CONNEXIONS™

The Interoperability Report

August 1989 | *Special Issue: Internetwork Routing* | Volume 3, No. 8

ConneXions — The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published monthly by Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, California 94040, USA. Phone: 415-941-3399. Fax: 415-949-1779.

ISSN 0894-5926

From the Editor

Anyone who is intimately involved in the design and operation of a computer network will tell you that *routing* represents one of their biggest headaches. A great deal of effort has gone into the development and refinement of routing architectures and algorithms for internetwork environments. In this issue we will explore some of these efforts and present routing from both a theoretical and practical perspective.

For completeness, we begin with a reprint of an overview article we ran in our June 1988 issue, entitled "Routing in an Internetwork Environment" by Ross Callon, Marianne Lepp and Varda Haimo.

Jose Joaquin Garcia-Luna-Aceves discusses methods for avoiding routing loops in a article starting on page 8. As you will discover, these topics are not free from academic controversy, in fact Dr. Garcia-Luna contradicts some of the statements made by Callon et al. As they say on Television: "Opposing views from responsible individuals are encouraged."

Much of the work to design new—or improve existing—routing algorithms is being done under the auspices of the Internet Engineering Task Force (IETF). In this issue we have articles describing two of these efforts; the OSPF Working Group and the Open Routing Working Group. The articles are by Rob Coltun and Marianne Lepp, respectively.

The *Adopt a Gateway* program was a community effort to save the Internet from terrible congestion problems in the days when all the core gateways were LSI-11s. Since then, the core gateways (or "routers" in OSI terminology) have been upgraded to BBN Butterflies. Bob Enger tells the story of how the adoption program came about and explains the infamous "Extra-Hop" problem.

Routing is certainly not unique to the TCP/IP world. Paul Tsuchiya and Rob Hagens describe the emerging OSI IS-IS and ES-IS routing architectures.

It should be pointed out that this issue contains information mostly on *intra-domain* routing. Inter-domain routing is another topic which we hope to cover in future issues.

For your benefit we have included a short glossary of commonly used routing terminology, see page 25.

Finally in this issue, we bring you some announcements about current Internet activities.

Routing in an Internetwork Environment

by Ross Callon*, Varda Haimo,
and Marianne Lepp, BBN Communications Corp.

Introduction and overview

The Internet provides a set of routing problems different from those of other network environments. The Internet is a *very* diverse environment under the control of multiple administrations, which utilizes equipment built by a wide variety of vendors. The design of an Internet routing algorithm must accommodate this fact. Because multiple administrations control the Internet, changes are hard to make and incorrect tuning can occur. In addition, the routing algorithm will be used in an environment in which the DoD and ISO internet protocols are used in parallel, with some gateways providing a dual IP function.

Autonomous Systems

Internet gateways are partitioned into *Autonomous Systems* [1]. This allows different parts of the Internet to be administratively separate, and allows gateways in different Autonomous Systems to use different routing protocols internally. For example, the BBN LSI-11 gateways use the GGP protocol, the newer BBN Butterfly gateways use a version of SPF, and many of the gateways in the NSFNET use RIP [2]. Gateways in different Autonomous Systems exchange reachability information via the *Exterior Gateway Protocol* (EGP) [1,3]. A similar approach is used in the OSI environment, where gateways (or "Intermediate Systems") as well as networks and hosts are split into "routing domains" [4-6].

This article considers issues in the design of a routing protocol for use within a single autonomous system or routing domain. Routing between domains is greatly complicated by the numerous administrative restrictions that may be imposed, and is beyond the scope of this article. The requirements for routing between multiple Autonomous Systems or routing domains in discussed in [7].

There are a number of issues that must be addressed in discussing the question of which techniques are appropriate for intra-Autonomous System routing in the Internet. Among the most important issues are the choice of algorithm, what levels of hierarchy the routing algorithm should have, how dynamic routing should be, the choice of metrics, and how routing information should be distributed. Each of these is discussed below.

Routing algorithms

There are two classes of commonly employed algorithms which choose shortest paths (with respect to a particular metric). We call *distance-vector* algorithms the class of algorithms in which a node sends to its neighbors a vector of distances (its routing table) and *link-state* algorithms those in which a node floods to all nodes the metric associated with its adjacent links. With distance-vector methods a node uses its neighbors' routing tables to determine routes. This apparently gives distance-vector algorithms an overhead advantage as compared to link-state algorithms. However, the vector of information shared between neighbors can be quite large. If a gateway has many neighbors this large update can be sent almost as many times as the small update from link state algorithms. This problem can be addressed by restricting the number of neighbors who receive updates to a number smaller than all a gateway's internet neighbors.

**This article is reprinted from our June 1988 issue, Mr. Callon is now with Digital Equipment Corporation.*

In addition, since each vector-state update often results in a chain of updates, this can constitute a de facto form of flooding. In general, however, distance-vector algorithms scale well.

Another apparent advantage for distance-vector routing is that only a small amount of CPU is used—one addition and one comparison for each neighbor. In contrast, the CPU used by link-state algorithms scales with the square of the number of nodes. This advantage is illusory, however, because the potentially slow convergence of distance-vector algorithms may force nodes to repeat these simple calculations many times before routes stabilize. Link-state algorithms use more CPU per computation, but they calculate new routes directly rather than converging towards the solution, thereby offsetting their apparent additional CPU requirements. In addition, link-state algorithms can reduce the amount of needed computation by doing incremental updates to the routing tables [8].

Routing loops

The most serious disadvantages to distance-vector based algorithms are that they can form *routing loops*, and they can be slow to converge, leading to route instability and increased overhead. There are a large number of fixes that have been implemented and/or proposed to reduce this problem [10-14]. However, these numerous fixes to distance-vector algorithms only reduce the seriousness of these problems—they do not eliminate them. In addition, since no global information is maintained, and since every node maintains a different set of information which is calculated from the information received from neighbors, errors in routing algorithm performance tend to be hard to identify. These convergence problems are discussed in greater detail in [15].

In contrast, link-state methods require that information about the state of the whole network be distributed across the net, resulting in a control traffic burden in large networks. The benefit of the link-state method is that the gateways maintain consistent views of the network, thus precluding problems such as looping and slow adjustment to changes in network conditions. In addition, since each node maintains an identical database, misbehaving nodes are somewhat easier to detect. Finally, link-state algorithms are easier to use with Type-of-Service Routing [17]. The relative advantages of each of these approaches is discussed in greater detail in [15, 16].

Hierarchical routing

The level of hierarchy of the Internet routing algorithm is an issue of exceptional importance given the rate of growth of the Internet. Both link-state and distance-vector algorithms will have problems in very large systems—link-state because of the amount of control traffic it engenders, and distance-vector because of its convergence problems. If Autonomous Systems grow very large, some level of hierarchy must be included in Internet routing to reduce the amount of information that must be processed at each node.

The main issues for hierarchical routing are determining where and how to define the levels of hierarchy, how to adjust to changes in topology (particularly changes that partition an area), how to compute routes, and how to propagate the routing information (including the information used to calculate routes, and/or the routes themselves).

continued on next page

Routing in an Internetwork Environment *(continued)*

There are a large number of ways to arrange networks into hierarchies. For example, "quasi-hierarchical" methods such as Kamoun-Kleinrock clustering [18] and Landmark Routing [19] use a hierarchy to summarize information, but combine detailed local information and summarized remote information into a single level of routing calculations. "Pure Hierarchical" methods separate a network or internet into multiple levels, and perform separate routing calculations at each level. Higher level routing calculations may compress lower level information in one of several ways. For example, higher level routing calculations may consider each underlying area to be a single node [20]. Alternatively, each area may be condensed into several nodes, such as one for each boundary with a neighboring area [21]. The manner in which information is hierarchically summarized has an effect on both the amount of routing information which is necessary to perform routing calculations, and on the quality and stability of the routes which are chosen.

Hierarchical routing algorithms can be used in systems which are very large, but there are costs. Since routing information is summarized, routes will not be as good as those found by a flat routing scheme. However, our work on SURAN included simulations which showed that the difference in routes is generally not significant [20]. Because of the problems with area determination and area partitions, hierarchical algorithms are more complicated than flat routing algorithms. There are also tradeoffs between the complexity of the algorithm and performance. For example, Kamoun-Kleinrock clustering or Landmark Routing will minimize the complexity of a hierarchical scheme, but requires the use of a distance vector routing algorithm which suffers from other problems.

Dynamic versus static routing algorithms

Routing algorithms can vary in responsiveness from using static paths to having paths change dynamically in response to the congestion state of resources. A static algorithm finds its routes and its back-up routes from tables provided to the gateway at configuration time. Such tables cannot be completely responsive in the face of failures since back-up routes may overlap on the failed resource. Furthermore, as the physical topology changes, every gateway must have its tables updated. Another serious problem with static routing is the difficulty of configuring routing tables correctly. As the internet grows the configuration problem rapidly worsens, thus introducing an increasing risk of mis-configuration. Static routing tables provide very stable routing, however.

A dynamic routing algorithm adapts to changing conditions. It can respond automatically to topology changes, or, more dynamically, it can also respond to congestion. However, dynamics can be a double-edged sword. Care must be taken to avoid routing oscillations and their attendant costs in control overhead, CPU, and congested resources. In a large and heterogeneous network environment such as the Internet, one must be very careful not to design a routing algorithm that responds to network conditions at a rate faster than the rate at which relevant information can reach decision points in the network.

For instance, a routing algorithm that responds to congestion after the congestion has already dissipated is not well-designed. These considerations are particularly important in the DoD Internet where control information must travel long distances, and where there is a large range of network types.

Another major advantage of dynamic routing schemes is that they allow some degree of automatic configuration of a network or internet. For example, dynamic routing may automatically incorporate new connections into its routes and automatically eliminate failed connections from its routes. The hierarchical scheme designed for the SURAN networks automatically configures packet radios into a layered hierarchy. Similarly, Landmark Routing allows for a high degree of self-configuration.

Choice of metrics

Metrics provide the criteria by which routing algorithms choose between alternate routes. As such, the appropriate choice of metrics for an internet routing algorithm is determined in large part by the requirements for the algorithm. Some possible choices are *hop count*, *administrative distance*, *capacity*, or *delay*. A metric that is fixed will yield stable routes, and will also permit routing to respond to outages. A metric that measures congestion will yield routes which change to avoid congestion, but this may cause routing oscillations if the metric is poorly chosen. The metric should be chosen in the context of the requirements for routing, and with considerations of consistent and effective algorithm design at the fore. The choice of routing metrics used to determine routes will have a major impact on both the quality of routes and the stability of the algorithm.

Distribution of routing information

Routing can be computed with a distributed computation or it can be computed at a centralized routing node (an "oracle") and distributed on request to gateways. With a distributed algorithm, routes can be computed jointly by each gateway so that only the destination is needed for forwarding gateways to identify the next hop. Alternatively, the source gateway can compute the route and source-route data.

With centralized routing the oracle makes all routing decisions and then announces them to the rest of the network. Depending upon the adaptivity of the routing algorithm, this central router needs information from the network on which to base its routing decisions. Thus it is necessary to have very good communication and sufficient capacity between the oracle and the rest of the network. The advantages are that routes chosen by a central router will be consistent, and that other network resources need not be expended on processing routing choices. The disadvantages are that the network is very vulnerable to congestion and communications problems at the central routing site or sites.

With distributed routing there are two main choices for determining paths. The source can determine routes, and then include them with the packet or install routes at forwarding gateways. The alternative is to allow each gateway to determine what the appropriate next hop should be, based on the destination of the packet (and perhaps on other factors such as packet type). The latter method is used by most single-path routing algorithms.

continued on next page

Routing in an Internetwork Environment *(continued)*

With multipath routing, consistency of routes is hard to maintain, making source routing attractive. Source routing would increase the size of the packet and use more memory and CPU in the gateways. Alternative encodings of the source route are possible, but this would require redesign of the Internet Protocols or definition of an IGP gateway-to-gateway header. The advantages of a source-routing scheme, such as no-looping, no repeated computation, and ease of encoding multiple paths in a multipath routing scheme, may be insufficient to justify the changes that it would entail.

References

[1] Linda J. Seamonson and Eric Rosen, "Stub Exterior Gateway Protocol," RFC 888, January 1984.

[2] C. Hedrick, "Routing Information Protocol," Rutgers University, RFC 1058, June 1988.

[3] Dave Mills, "Exterior Gateway Protocol Formal Specification," RFC 904, April 1984.

[4] "OSI Routing Framework," ISO TC97/SC6/N4616, June 1987.

[5] Tassos Nakassis, "Basic Issues of Routing Between Routing Domains," NBS, September 1987.

[6] "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol," source USA, ISOTC97/SC6/N4945, October 1987.

[7] Ross Callon (editor) "Issues in Inter-Autonomous Systems Routing," (Submitted to the RFC Editor).

[8] J. McQuillan, I. Richer, and E. Rosen, "ARPANET Routing Algorithm Improvements, First Semiannual Technical Report," BBN Report 3803, April 1978.

[9] E. Rosen, J. Mayersohn, P. Sevcik, G. Williams, and R. Attar, "ARPANET Routing Algorithm Improvements, Volume 1," BBN Report 4473, August 1980.

[10] J. Jaffe and M. Moss, "A Responsive Distributed Routing Algorithm for Computer Networks," IEEE Transactions on Communications, vol. COM-30, pp. 1758-1762, July 1982.

[11] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," EE PUB No. 313, Dept. of EE, Technion—Israel Institute of Technology, Haifa, Israel (1978).

[12] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," IEEE Trans. Comm. Com-27, No. 9 (1979) pp 1280-1287.

[13] S. Toueg, "A minimum-hop path failsafe and loop-free distributed algorithm," IBM Research Report RC 8530 (1980).

[14] J. Garcia-Luna-Aceves, "A New Minimum-Hop Routing Algorithm," SRI International, 1987.

[15] R. Callon, "A Comparison of 'Link State' and 'Distance Vector' Routing Algorithms," (Unpublished).

[16] L. Bosack, "A Further Comparison of 'Link State' and 'Distance Vector' Routing Algorithms," cisco Systems, February 1988.

[17] M. Lepp, I. Loobeek, and S. Cohn, "Type-of-Service Routing: Preliminary Design," BBN Report 6195, June 1986.

[18] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks," Computer Networks, Vol 1, No. 3, January 1977.

[19] Paul F. Tsuchiya, "Landmark Routing: Architecture, Algorithms, and Issues," MTR-87W00174, September 1987.

[20] R. Callon and G. Lauer, "Hierarchical Routing for Packet Radio Networks," BBN Report 5945, SRNTN No. 31, June 1985.

[21] A. Khanna, "Large Network Routing Study: Final Report," BBN Report 6267, October 1986.

This work was supported by the Defense Advanced Research Project Agency, under contract number F29601-87-C-0086. Distribution is unlimited

ROSS CALLON received his B.Sc. in Mathematics from the Massachusetts Institute of Technology in 1973, and his M.Sc. in Operations Research from Stanford University in 1977. He was with BBN from 1980 to 1988, were he was the Internet Architect for the Gateway Development group. He is currently in the Distributed Systems Architecture Group at Digital Equipment Corporation in Littleton, MA. He is concerned with a variety of design issues relating to the internet architecture, including internet routing, and interoperability of TCP/IP and OSI. Mr. Callon is co-chair of the Internet Engineering Task Force OSI Interoperability Working Group.

VARDA HAIMO received her Ph.D. in Applied Mathematics from Harvard University in 1984. She spent a year as a Postdoctoral Research Fellow at the Harvard Business School, after which she came to BBN Communications. At BBN Varda manages the Algorithm Development, Simulation and Modelling group in the Network Analysis Department. She has worked extensively in the area of routing in packet-switched networks: participating in the design, modelling and simulation of a new Type-Of-Service routing algorithm for the DDN.

MARIANNE LEPP received her Ph.D in mathematics from the University of Wisconsin, Madison in 1975, and held academic positions at North Carolina State University and Worcester Polytechnic Institute. For the past 4 years she has worked at BBN Communications on a variety of networking projects including network design and design tool and algorithm development, performance analysis, and protocol development for both BBN packet-switched networks and the DoD Internet. Among her key projects was the design of a Type-Of-Service routing algorithm with loadsharing for BBN packet-switched networks. In addition, she is a member of the Internet Engineering Task Force, chaired the EGP3 working group and chairs the Open Routing working group, both part of the IETF.

Loop-Free Internet Routing and Related Issues

by **J.J. Garcia-Luna-Aceves,**
Network Information Systems Center,
SRI International

Introduction

The shortest-path routing algorithms used today in computer networks can be classified as distance-vector or link-state algorithms. In a *distance-vector algorithm*, a node knows the length of the shortest path from each neighbor node to every network destination, and uses this information to compute the shortest path and next node in the path to each destination. A node sends update messages to its neighbors—and only to them. Each update message contains a vector of one or more entries; typically, each entry specifies the distance to a given destination. The simplest distance-vector algorithm is a decentralized implementation of the Bellman-Ford algorithm for shortest-path computation [7, 22]. In contrast, in a *link-state algorithm* a node must know the entire network topology, or at least receive that information, to compute the shortest path to each network destination. Each node broadcasts update messages, each containing the state of each of the node's adjacent links, to every other node in the network. The new Arpanet routing algorithm [19] is a well-known link-state algorithm.

Background

Over the past few years, there has been a great deal of debate as to which type of routing algorithm is better suited for networks with dynamic topologies and varying traffic flows. As Schwartz points out [22], deciding between one or the other is highly dependent on the specific network in which the algorithm will operate. The primary concern with the distance-vector algorithms that have been used in the past is the formation of long-lasting *routing-table loops*; a routing-table loop is a path specified in the nodes' routing tables at a particular point in time that visits the same node more than once before reaching the intended destination. Recently, Callon, Haimo, and Lepp [3] have stated that freedom of routing-table loops cannot be achieved using distance-vector algorithms. However, Merlin and Segall [20], Dijkstra and Scholten [6], Chandy and Misra [1], and the author [9] have all verified mechanisms for routing-table loop freedom that are applicable to distance-vector algorithms.

On the other hand, link-state algorithms are free of long-lasting routing-table loops. However, they need to maintain an up-to-date version of the entire network topology at every node, which may constitute excessive storage and communication overhead in a large network (e.g., see the work by Seeger and Khanna [23]). It is also interesting to note that the link-state algorithms proposed or implemented to date do not eliminate the creation of *temporary* routing-table loops, which can be created whenever two or more nodes recompute shortest paths by using inconsistent views of the network or internet topology.

Loops

Whether link states or distance vectors are used, the existence of routing-table loops, even when these are temporary, is a detriment to the overall network performance. In this article, we introduce a unified approach to the solution of the looping problems of distributed routing algorithms. This approach is independent of whether distance vectors or link states are used, the metric used to measure internodal distances, or the cost function used to compute shortest paths.

We treat the shortest-path routing problem as one of *diffusing computations*—a concept that was first proposed by Dijkstra and Scholten [6] for detecting the termination of distributed computations. Although shortest-path routing in relatively small networks (e.g., the size of the Arpanet) can be accomplished very effectively using any of several existing routing algorithms, our approach has interesting implications for the development of new routing and network reachability protocols for large internets of arbitrary topologies.

In the rest of this article, we refer to routing-table loops simply as *loops*. The loop freedom in which we are interested concerns the entries specified in the nodal routing tables.

Split-horizon

In the past, there have been a number of partial solutions to the routing-table looping problems of distance-vector algorithms. The most widely known of these is the *split-horizon* technique [4]. According to this technique, nodes exchange update messages containing the shortest distances and their successors (i.e., next hops) to one or more network destinations. A node never chooses as its successor toward a destination a neighbor that has reported the node as its own successor to the same destination.

Stern [25] has proposed an algorithm that uses the split-horizon technique, whereby a node sets its distance to infinity when its perceived minimum distance to a destination increases. The author [8] has proposed an algorithm, similar to Stern's, in which the split-horizon technique is used and in which a node chooses, as the successor to a network destination, a neighbor that has reported *either* a decreasing distance to that destination *or* a constant distance and a constant successor.

Hold-downs

A number of authors have proposed the use of *hold-downs*. With this technique, when a node receives an update from its current successor indicating a distance increase, it must wait for a fixed period of time before making any changes to its routing table.

Naylor [21] has proposed a loop-free algorithm that uses the split-horizon technique in combination with a loop-check message that is sent to a destination node along the path starting with a given neighbor node when a node needs to update its routing table regarding that destination and suspects that a multinode loop can be established.

Unfortunately, as discussed by the author elsewhere [11], none of the above techniques solves all the looping problems of the decentralized implementation of the Bellman-Ford algorithm.

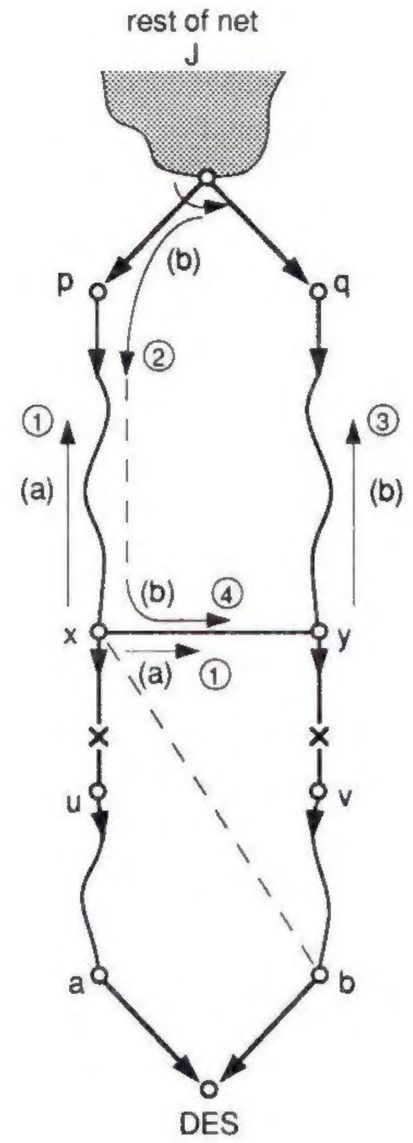
Juncture nodes

Tsuchiya [26, 27] has proposed (without proof) a routing algorithm that establishes a labeling mechanism on the shortest paths used by nodes. A node *A* adjacent to a destination *DES* labels the path with its own identifier. The label is assumed by all nodes that use *A* in their preferred shortest paths to *DES*, until a "juncture node" *J* [26] (one with more than one shortest path with a unique label to *DES*) is reached. Node *J* labels its primary shortest path to *DES* with its own label, and nonjuncture nodes that have *J* in their paths to *DES* assume node *J*'s label.

continued on next page

Loop-Free Internet Routing *(continued)*

Tsuchiya's algorithm includes a mechanism for nodes to determine whether or not they are juncture nodes; the hope is that a distance increase in the path from a destination to a juncture node will create a chain of updates that reach the juncture node, which will be able to choose an alternative path with no looping. Unfortunately, as Figure 1 illustrates, this mechanism appears to fail in solving the counting-to-infinity problem when multiple failures occur.



(1) Bad news propagates to **J** from **x**

(2) **J** uses **q** for alternate path to **DES**

(3) Bad news propagates to **J** from **y**

(4) Erroneous fix propagates to **y** from **J**

Figure 1: Looping with Juncture Nodes

In the figure, node J is a juncture node with two possible paths (through p and q) with different labels (a and b). When links (x, u) and (y, v) fail, node J can receive distance-increase news from node p and use q as its alternative successor to DES, while distance-increase news from y propagate toward J. Because node y cannot tell from the label reported by node x whether or not x has a valid path (which would be as shown in dashed lines), and because no juncture nodes (which could relabel the path) are left in the loop in which x, y, and J participate, counting-to-infinity can occur.

Other solutions

Shin and Chen have proposed a strategy that attempts to eliminate all routing-table loops by including the complete path from source to destination both in update messages and in routing-table entries [24]. It is easy to show by example that this technique does not guarantee freedom of temporary loops under multiple, topological changes. The author has recently proposed another loop-free, distance-vector algorithm for minimum-hop routing [11] that uses Shin and Chen's strategy to eliminate all types of loops, but this algorithm is not practical for networks with links of arbitrary cost.

A number of algorithms have been proposed in the past to ensure loop freedom in shortest-path routing by means of internodal coordination. The algorithm by Merlin and Segall [20] is based on the propagation of information with feedback along a tree rooted at a network destination and spanning all network nodes.

A major limitation of this algorithm is that the destination node (or root of the tree), or the node that first detects a change in distance to that destination, must start each *generation* of the propagation of updates, where each generation reaches an additional hop of the tree. This creates substantial communication overhead [22].

Diffusing computations

An alternative approach to the propagation of information with feedback used by Merlin and Segall is to view distributed loop-free routing as a problem of *diffusing computations*. Dijkstra and Scholten [6] introduced this technique to check the termination of a distributed computation, such that the process starting a computation is informed when it is completed and such that there are no false terminations. The diffusing computation grows by sending *queries* and shrinks by receiving *replies* among the nodes participating in the diffusing computation. This algorithm assumes the existence of a link-level protocol that provides the same services available between neighbor nodes in a network, or gateways in an internet.

Successor tree

To apply this concept to routing, we first observe that, regardless of whether distance vectors or link states are used, each nodal routing table must contain the length and successor (i.e., next node) of the shortest path to each destination. We consider that each link has two lengths (or costs) associated with it—one for each direction—and that any link of the network exists in both directions at any one time. For the moment, we can assume that each node maintains a *topology table* with sufficient information to compute the values of the node's routing table. When all routing tables are correct, for each network destination, the successor entries in the routing tables of all nodes define a tree, which we call the *successor tree* for destination j. A successor tree has to be recomputed each time topological changes affect some of its links or nodes.

Computation

Using Dijkstra and Scholten's algorithm (DSA), a single successor tree computation can be distributed among nodes along the successor tree by using queries that contain the news about the topological change; the successor tree computation then shrinks along the same tree with the reception of replies to those queries, which specify that the nodes sending the replies have modified their routing tables according to the change(s) reported in queries. Initially, all nodes are *passive*. When a node detects a change in the status or cost of an adjacent link, a successor-tree computation commences with the node going into *active* state by sending a query to its neighbors.

After processing a query from its successor toward the destination, a passive node forwards the query to its neighbors uptree in the successor tree and goes into active state. A node in active state cannot change its successor to the destination until it receives all the replies to its query; when this happens, the node goes back to passive state and computes a new successor and distance. Accordingly, all the neighbors of an active node must receive the new distance from that node to the destination and reply with their own new distances before the node can recompute its distance and successor to the destination. This means that a node can never choose a neighbor uptree in the successor tree when it must change its successor to the destination. Chandy and Misra proposed an adaptation of DSA very similar to the one summarized above [1].

continued on next page

Loop-Free Internet Routing *(continued)*

Independently of the work by Chandy and Misra or Dijkstra and Scholten, Jaffe and Moss [15] showed that no routing loops can occur after a link addition or a link-cost decrease. Using this result, they proposed a loop-free distance-vector algorithm that supports multiple diffusing computations for the same network destination, and that requires no internodal coordination unless link costs increase or resources fail in the network. In the absence of link-cost increases or resource failures, the Jaffe-Moss algorithm behaves like the Bellman-Ford routing algorithm, that is, no diffusing computation is used. However, when a node detects an increase in the cost of a link, it starts the computation of a new successor tree for each destination for which the node's distance was affected by the change. The new successor tree is computed using an algorithm similar to the adaptation of DSA described previously. Multiple concurrent successor-tree computations are supported by means of bit vectors, maintained at each node, which specify, for each neighbor and for each destination, how many queries need to be answered by the neighbor node and which one of such queries was originated by the node maintaining the bit vector.

Feasible successor

Finally, the author extended the Jaffe-Moss algorithm by reducing the degree of internodal coordination even further [9]. According to this algorithm, when a node needs to update its routing table for a given destination j after it processes an update message from a neighbor, or detects a change in the cost or availability of a link, the node tries to obtain a *feasible successor* with the shortest distance to node j. From the standpoint of node i, a feasible successor toward node j is a neighbor node that has reported a distance to the destination that is at most as large as the distance reported by node i's current successor before node i decided to update its routing table. When feasible successors are found, the algorithm behaves as the Bellman-Ford algorithm, or else, if a node cannot find a feasible successor with the shortest distance to node j, the node starts a successor tree computation for node j and sends a query to all its neighbors. The node cannot change its successor to node j until it has been told by all its neighbors that they have processed its query. Bit vectors are used to support multiple, concurrent successor tree computations.

Contrary to Callon's description [2] of the Jaffe-Moss algorithm, diffusing computations are very different from hold-downs. A hold-down is a unilateral action taken by a node without knowledge of the effect it will have in preventing loops. A diffusing computation is an algorithm implemented at each node to ensure the detection of the correct termination of a distributed computation. Callon [2] has also stated that nodes are not able to forward traffic while they wait to become passive (according to our description above); however, the only case in which the statement is true is after resource failures. Furthermore, the Garcia-Luna algorithm [9] allows the use of alternative paths after resource failures when feasible successors are found.

A unified approach to Loop Freedom

In this section, we present a new algorithm that generalizes the algorithms based on diffusing computations outlined in the previous section by separating the mechanism used for diffusing the computation of a successor tree from the contents of the nodal routing tables and topology tables.

We call such an algorithm the *distributed updating algorithm* or DUA. We describe DUA at a level of detail that is independent of implementation considerations. We assume that either of two well-known algorithms is used at each node to compute shortest paths once topology information is gathered using DUA—Bellman and Ford's distance-vector algorithm [7] or Dijkstra's link-state algorithm [5]. A more detailed description of the new algorithm is provided in a companion publication, in which we verify that DUA is loop free at every instant and correct [12].

Messages

There are three types of messages in DUA—*updates*, *queries*, and *replies*. Updates and queries contain an update list of one or more entries, each of whose contents is dependent on whether link states or distance vectors are used. A reply is sent in response to *all* the update-list entries in a query. Messages can be exchanged periodically or on an event-driven basis. When distance vectors are used, an update-list entry contains, as a minimum, the identifier of a destination and the current shortest distance to it. In contrast, when link states are used, an update-list entry consists of the identifier of a link and its cost.

Data Structures

Each node in the network maintains a *routing table*, a *link-state table*, a *reply-status table*, a *query-origin table*, and a *topology table*.

The routing table of node i has an entry for each node in the network. The entry corresponding to node j consists of the identifier of node j, the length of the current distance from node i to node j (denoted by D^i_j), and the successor in the path currently chosen by node i toward node j (denoted by S^i_j)

Each entry of the link-state table of node i corresponds to the cost of a link adjacent to the node. The cost of an inactive link is set to infinity. For each destination j and for each neighbor k of node i, the reply-status table of node i specifies a state flag (s^i_{jk}) that indicates whether or not node i is waiting for node k's reply to a query. For each destination j, the query-origin table of node i specifies the identifier of node j and a one-bit flag (denoted by r^i_j) that states whether or not node i originated the query that made the node become active.

When distance vectors are used, each entry of the topology table consists of the identifier of a destination j and the length of the current distance from node k to node j reported by node k, for each neighbor k of node i, denoted by D^i_{jk}.

In contrast, when link states are used, the topology table consists of a link-cost matrix that contains an entry for each pair of network nodes with the cost or weight of a link; the cost of a link that does not exist is assumed to be infinity. We note that node i can obtain D^i_{jk} for any neighbor k of node i directly from the link-cost matrix.

Operation of DUA

In the following description, we always refer to a node's state (active or passive) with respect to destination j, and refer to node j, distance to node j, successor tree for node j, and successor toward node j simply by *destination*, *distance*, *successor tree*, and *successor*, respectively. A node can be either active or passive with respect to node j. A node is active when it is waiting for replies from its neighbors referring to node j, and is passive otherwise.

continued on next page

Loop-Free Internet Routing *(continued)*

If node *i* is active, it cannot modify its successor; furthermore, node *i*'s distance must be set equal to the sum of the distance reported by the current successor, plus the cost of the link with that neighbor.

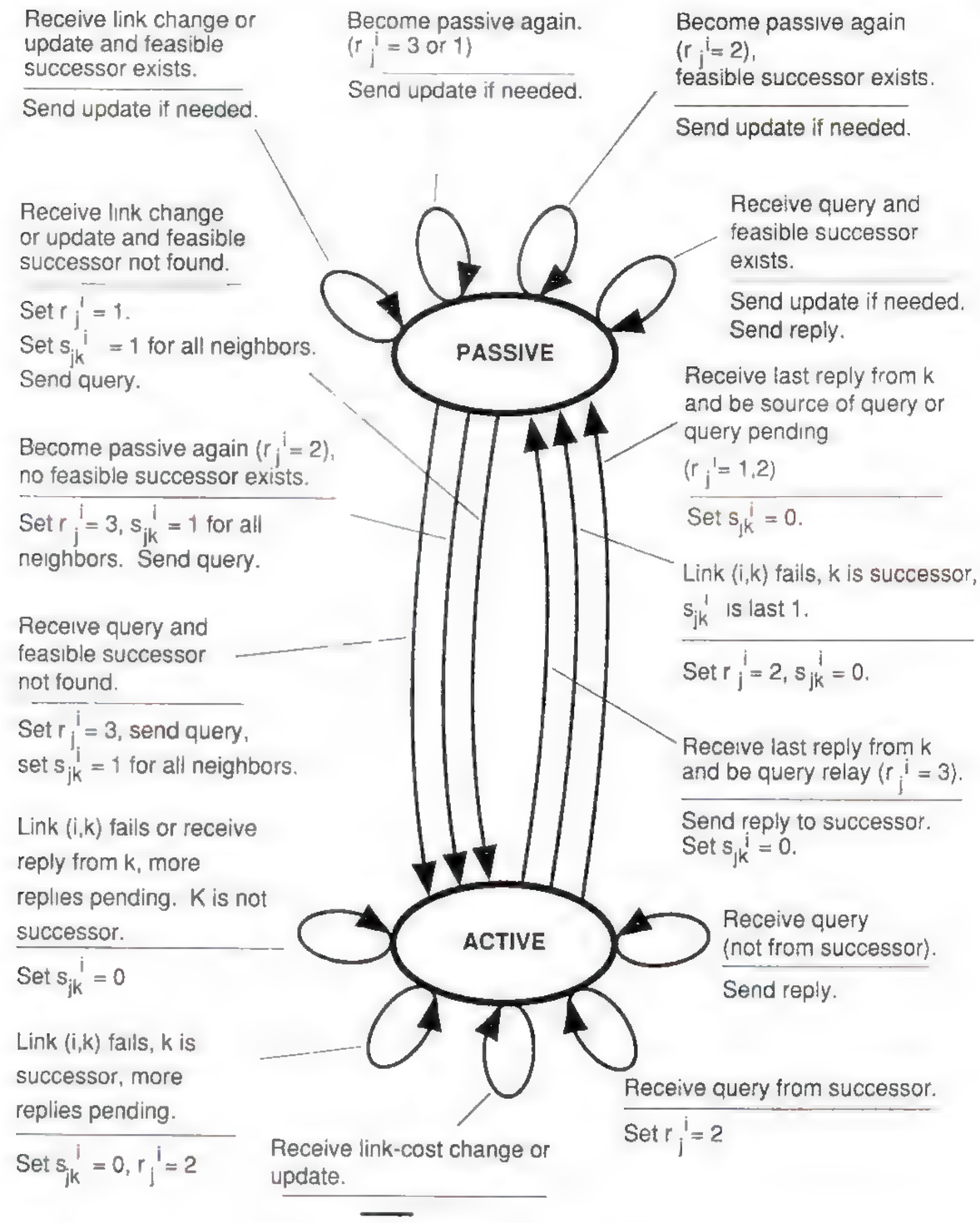


Figure 2: Active and Passive States in DUA

Figure 2 shows the basics of DUA; in the figure, we indicate only the information received at a node and the information sent by the node according to the state in which it is and the information it receives. Node *i*'s shortest distance equals the minimum, for all neighbors that are feasible successors, of the sum of the distance reported by a neighbor plus the cost of the link with that neighbor. This entry is set to infinity when node *i* determines that node *j* is unreachable. A feasible successor is defined as in the previous section.

When a passive node receives an input event (an update, a query, or a change in the cost or status of a link), it updates its topology table and, in the case of a link change, its link-state table. Using the new topology-table information, the node attempts to find a feasible successor. If a feasible successor is found, the node updates its routing table, remains in the passive state and, if needed, sends out an update to all its neighbors with the new topology or routing information (see Figure 2). If the input event was a query from a neighbor, the node sends a reply to that neighbor. A node that sends an update expects no replies and is free to process other input events. If a passive node *i* cannot find a feasible successor after processing an input event, then the node becomes active (see Figure 2) by setting $s^i_{jk} = 1$ for each neighbor *k* of node *i*. Node *i* resets s^i_{jk} to 0 upon reception of node *k*'s reply.

An active node that receives an update, or detects a change in the cost or status of a link, first updates its topology table (and its link-state table if needed), and then it updates its routing table without changing its successor. If needed, the node sends out an update to all its neighbors with the new topology or routing information. Notice, however, that the node stays active (see Figure 2). If an active node receives a query, it follows the same previous steps and, in addition, sends a reply to the query received. When node i is active and receives a reply from node k, it sets $s^i_{jk} = 0$.

Once active, node i stays that way until it receives the replies from all its neighbors, in which case every node uptree in the successor tree has incorporated the change in link cost or status in its topology table. Furthermore, it cannot send another query until it becomes passive; accordingly, only one diffusing computation can exist per subtree of the same successor tree. When node i becomes passive again (see Figure 2), it recomputes the elements of its routing-table entry, updates them if necessary and, if any change was made to its successor or distance, follows the same steps of a passive node that processes an input event.

Updates are distributed differently across the network, depending on whether distance vectors or link states are used. In the first case, an update needs to traverse a single hop when the routing table of the sending node is modified. In the second case, an update must be propagated to all network nodes. DUA works correctly with either type of operation. [12]

An active node i knows whether or not it was the originator or a simple relay of a query, by means of the query-origin flag [12], which is set to $r^i_j = 1$ if node i was the origin of the query. When link (i, S^i_j) fails, node i assumes that it was the source of any outstanding query and that its successor has replied to it.

When distance vectors are used, a diffusing successor-tree computation for destination j is completely independent from any other destination's successor-tree computation. A node detecting a change in the status or cost of a link may have to send an update with its shortest distances to destinations for which it has feasible successors, and a query regarding destinations for which the node has no feasible successors. On the other hand, when link states are used, the information exchanged in updates and queries refers to link costs only. Therefore, when a node receives a query with an update entry for link (x, y), it cannot determine the destinations for which the sending node is active. Because of this, a passive node that becomes active for at least one destination and sends out a query must act as if it were active for all destinations when it receives subsequent queries and updates, until it becomes passive again for all destinations. When the node receives the replies to its query and becomes idle, it must recompute all its routing-table entries.

Applications to internet routing

The preceding description of DUA illustrates the fact that loop-free routing can be accomplished independently of the type of information stored in nodal tables. Several enhancements can and should be made to DUA to make it more efficient for either link states or distance vectors [12]. Many issues need to be resolved to apply the mechanisms used in DUA to internet routing protocols; we address a few of them here.

continued on next page

Loop-Free Internet Routing *(continued)*

Loop freedom by itself seems less attractive for link-state algorithms, given the additional overhead needed for internodal coordination. However, loop freedom is very important for a distance-vector algorithm, because it impacts its speed of convergence, and is particularly important for hierarchical routing. Hierarchical routing becomes a necessity in a large internet to reduce routing overhead. As pointed out by Callon, Haimo, and Lepp [3], and the author [10], link-state algorithms incur substantial communication overhead in hierarchical routing. On the other hand, because a loop-free distance-vector algorithm converges much faster than traditional distance-vector algorithms [9], loop-free, hierarchical distance-vector routing is a very efficient approach for routing in a large internet insofar as routing overhead is concerned. Furthermore, several hierarchical routing methods proposed to date [10, 17, 27] can be used together with the loop-freedom mechanisms of DUA.

Source routing [14], multicasting, and routing through multiple paths from source to destination are all important in an internet, and can be accomplished using either distance vectors or link states.

Providing different types of service in an internet is important, and can be accomplished using either link states [18] or distance vectors [16]. The choice between the two approaches depends on the overhead of the protocol and the number of services to be provided. If only a few types of service are provided, as in the DDN routing algorithm [13], there is no big difference between using distance vectors or link states. However, storage requirements for a distance-vector algorithm grow proportional to the combinations of type-of-service parameters, while they grow proportional to the number of type-of-service parameters in a link-state algorithm.

Finally, DUA ensures that routing or topology information is distributed among nodes in such a way that nodes always detect correctly when the computation of a new route terminates, independently of the type of information used to calculate those routes. Accordingly, the mechanisms used in DUA are applicable to reachability algorithms, in which internodal distances are boolean values or are assigned based on administrative concerns.

References

[1] K.M. Chandy and J. Misra, "Distributed Computation on Graphs: Shortest Path Algorithms," in *Communications of the ACM*, Vol. 25, No. 11, November 1982, pp. 833–837.

[2] R. Callon, "A Comparison of 'Link State' and 'Distance Vector' Routing Algorithms," (Unpublished).

[3] R. Callon, V. Haimo, and M. Lepp, "Routing in an Internetwork Environment," in *ConneXions*, Vol. 2, No. 6, June 1988, pp. 4–9 and reprinted in this issue of *ConneXions*, pp. 2-7.

[4] T. Cegrell, "A Routing Procedure for the TIDAS Message-Switching Network," in *IEEE Transactions on Communications*, Vol. COM-23, No. 6, June 1975, pp. 575–585.

[5] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs," in *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[6] E.W. Dijkstra and C.S. Scholten, "Termination Detection for Diffusing Computations," in *Information Processing Letters*, Vol. 11, No. 1, August 1980, pp. 1–4.

[7] L.R. Ford and D.R. Fulkerson, "Flows in Networks," Princeton University Press, 1962.

[8] J.J. Garcia-Luna-Aceves, "A New Minimum-Hop Routing Algorithm," in Proceedings of IEEE INFOCOM '87, 1987.

[9] J.J. Garcia-Luna-Aceves, "A Distributed Loop-Free, Shortest-Path Routing Algorithm," in Proceedings of IEEE INFOCOM '88, 1988.

[10] J.J. Garcia-Luna-Aceves, "Routing Management in Very Large Scale Networks," in *Future Generation Computer Systems*, Vol. 4, No. 2, September 1988, pp. 81–94.

[11] J.J. Garcia-Luna-Aceves, "A Minimum-Hop Routing Algorithm Based on Distributed Information," in *Computer Networks and ISDN Systems*, Vol. 16, pp. 367-382, May 1989.

[12] J.J. Garcia-Luna-Aceves, "A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States," Proceedings of ACM SIGCOMM '89, Austin, Texas, September 1989.

[13] M. Lepp, I. Loobeek, and S. Cohn, "Type-of-Service Routing: Preliminary Design," BBN Report 6195, June 1986.

[14] J. Hagouel, "Issues in Routing for Large and Dynamic Networks," IBM Research Report RC 9942 (No. 44055), IBM Thomas J. Watson Research Center, April 1983.

[15] J.M. Jaffe and F.M. Moss, "A Responsive Routing Algorithm for Computer Networks," in *IEEE Transactions on Communications*, Vol. COM-30, No. 7, July 1982, pp. 1758–1762.

[16] J. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," in *Networks*, Vol. 14, No. 1, 1984, pp. 95–116.

[17] F. Kamoun, "Design Considerations for Large Computer Communication Networks," UCLA-ENG-7642, Computer Science Department, 1976.

[18] R. Kung and N. Shacham, "An Algorithm for the Shortest Path Under Multiple Constraints," in Proceedings of GLOBECOM '84, November 1984, pp. 355–359.

[19] J. McQuillan, et al., "The New Routing Algorithm for the Arpanet," in *IEEE Transactions on Communications*, Vol. COM-28, May 1980.

[20] P.M. Merlin and A. Segall, "A Failsafe Distributed Routing Protocol," in *IEEE Transactions on Communications*, Vol. COM-27, No. 9, September 1979, pp. 1280–1288.

[21] W.E. Naylor "A Loop-Free Adaptive Routing Algorithm for Packet Switched Networks," Proceedings of the Fourth Data Communications Symposium, October 1975, pp. 7.9–7.15.

continued on next page

Loop-Free Internet Routing *(continued)*

[22] M. Schwartz, "Telecommunication Networks: Protocols, Modeling and Analysis," Chapter 6, Addison-Wesley Publishing Co., 1986.

[23] J. Seeger and A. Khanna, "Reducing Routing Overhead in a Growing DDN," in Proceedings MILCOM '86, 1986, pp. 15.3.1–15.3.13.

[24] K.G. Shin and M. Chen, "Performance Analysis of Distributed Routing Strategies Free of Ping-Pong-Type Looping," in *IEEE Transactions on Computers*, Vol. COMP-36, No. 2, Feb. 1987, pp. 129–137.

[25] T.E. Stern, "An Improved Routing Algorithm for Distributed Computer Networks," in *IEEE International Symposium on Circuits and Systems*, Workshop on Large-Scale Networks and Systems, April 1980.

[26] P. Tsuchiya, "Landmark Routing: Architecture, Algorithms, and Issues," Draft Report MTR-87W00174, The MITRE Corporation, September 1987.

[27] P. Tsuchiya, "The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks," in Proceedings of ACM SIGCOMM '88, August 1988, pp. 35–42.

This work was supported by SRI International IR&D funds and the U.S. Army Research Office under contract DAAL03-88-K-0054, and by Rome Air Development Center and the Defense Advanced Research Projects Agency under Contract F30602-89-C-0015.

JOSE JOAQUIN GARCIA-LUNA-ACEVES received the B.S. degree in Electrical Engineering from the Universidad Iberoamericana, Mexico City, Mexico, in 1977, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Hawaii, Honolulu, Hawaii, in 1980 and 1983, respectively. Dr. Garcia-Luna joined SRI International in 1982, initially as an SRI International Fellow, and is currently Deputy Director of the Network Information Systems Center of SRI International. He was Assistant Professor at the Universidad Iberoamericana from 1977 to 1978. His current research interests include the analysis and design of distributed network algorithms and multimedia information systems. Dr. Garcia-Luna was General Chair of the ACM SIGCOMM '88 Symposium and Program Chair of the ACM SIGCOMM '87 and '86 symposia. He is a member of the ACM, IEEE, AAAS, IETF, and UITF.

A quote on Routing

"In general, routing is a *problem*." —*Marshall Rose*

The quote is taken from his his soon-to-be-published book on OSI called *The Open Book: A Practical Prespective on Open Systems Interconnection,* Publisher: Prentice-Hall, ISBN 0-13-643-016-3*)*.

OSPF: An internet routing protocol

by Rob Coltun, University of Maryland

Introduction

OSPF is an IP routing protocol classified in internet terminology as an *Internal Gateway routing Protocol* (IGP). An IGP runs within an Autonomous System (AS); an AS being defined as a collection of routers all running a single IGP. OSPF is an acronym for *Open Shortest Path First* (shortened from OSPFIGP). It is an open routing protocol in that the protocol specification is openly available; it is not proprietary to any one vendor. It is an SPF routing protocol because it uses SPF technology.

History

The growth in networking over the past few years has pushed the currently available open IGPs (*RIP*, *Hello*) past their limits. OSPF came out of the work of the OSPFIGP working group of the Internet Engineering Task Force co-chaired by John Moy of Proteon and Mike Petry of the University of Maryland. John Moy wrote the specification. The working group was formed in 1988 to develop an SPF-based IGP. The foundation research used to design OSPF was 1) BBN's development of an SPF algorithm for the Arpanet in 1978, 2) Radia Perlman's research on fault-tolerant broadcast of routing information in 1983, 3) BBN's work on area routing in 1986, and 4) ISO's IS-IS Intra-domain Routing Information Exchange Protocol.

OSPF features

Some features of OSPF are:

- OSPF combines fast response to topological changes with minimization of routing traffic.
- An OSPF router can calculate more than one minimum cost path to a destination if more than one path exists having the same minimum cost (equal-cost multipath routing). This will allow a router to do load balancing.
- OSPF is resistant to routing loops.
- There is a 16-bit configurable metric (large enough to allow for expansion) for each type of service per interface; OSPF is designed to compute separate routes for each type of service.
- OSPF is tailored for the IP environment. For example, subnet masks can be attached to routes thereby allowing variable length subnet masks in implementations.
- OSPF routing exchanges are authenticated.
- On broadcast networks, OSPF uses IP multicast instead of broadcast for its routing exchanges. An OSPF router is a "good neighbor" to hosts and other routers to which it is physically connected on a broadcast network by not involving them in routing exchanges unless they are participating in the OSPF protocol.
- OSPF packets are designed for efficient processing. For example, fields observe byte boundaries (as well as 4-byte boundaries) and fields that serve the same function (such as age and checksum fields) in different types of packet headers are located in the same byte position.

continued on next page

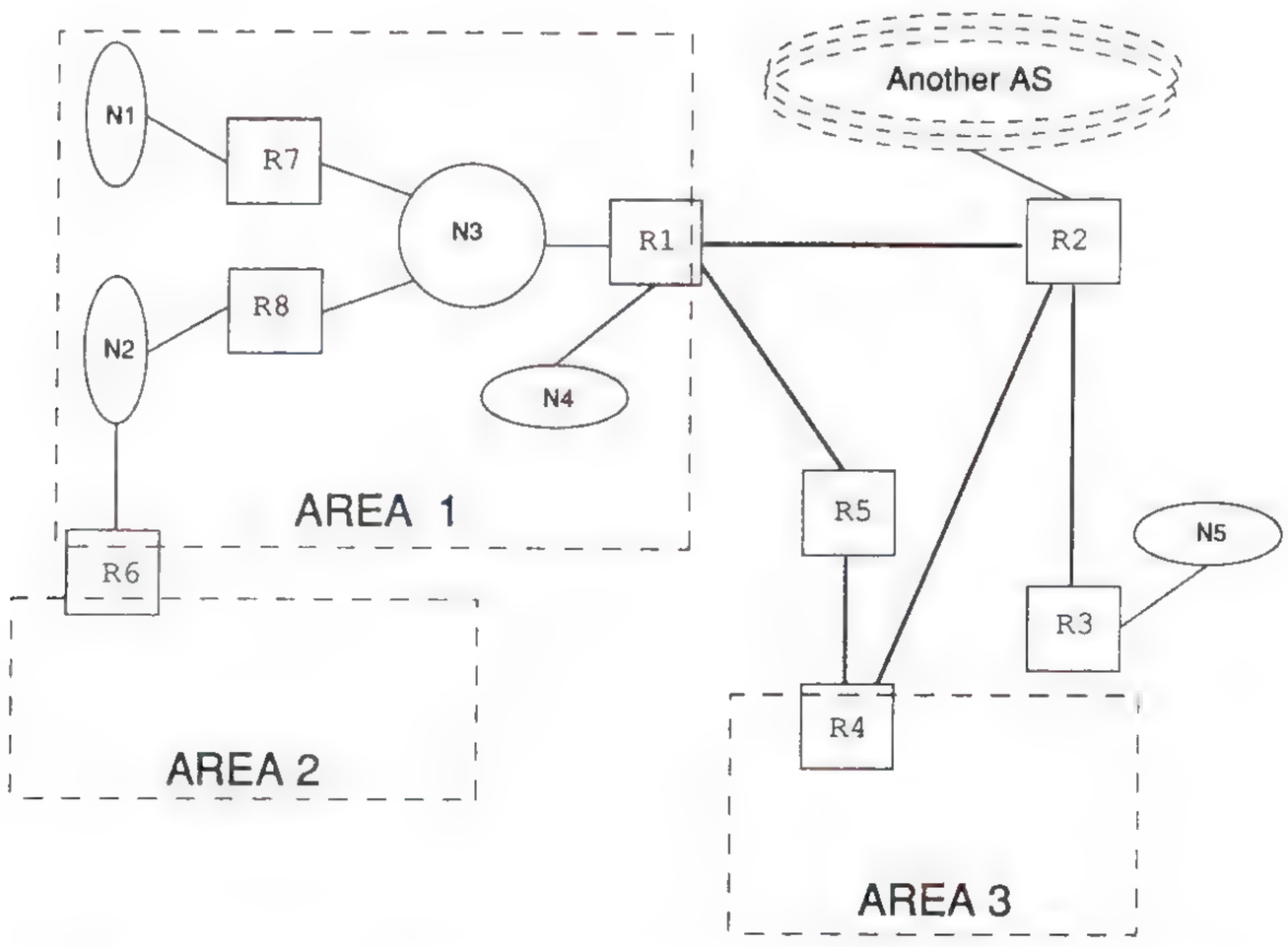
OSPF: An internet routing protocol *(continued)*

- OSPF supports an area routing scheme. This allows another level of information hiding and reduction, and protects routing within an area from outside interference.
- OSPF distributes externally derived routes (like EGP routes) independently from internally derived routing information.

SPF

The three important features of an SPF routing protocol are 1) all routers have identical routing databases, 2) each router's database describes the complete topology (which routers are connected to which networks) of the router's domain, and 3) each router uses its database to derive the set of shortest paths to all destinations from which it builds its routing table.

In general, an SPF protocol works as follows. Each router periodically sends out a description of its connections to its neighbors (the state of its links). (Routers are neighbors if they are directly connected via a network such as a point-to-point link (e.g., R2 and R3 in our example) or a multi-access network (e.g., R1, R7, and R8 in our example.) This description, often called a *Link-State Advertisement* (LSA), includes the configured cost of the connection. The LSA is flooded throughout the router's domain. (A routers domain could be an area or the entire AS (see the sections on OSPF Area Routing, The OSPF Backbone, and External Routes). Each router in the domain maintains an identical synchronized copy of a database composed of this link-state information. This database describes both the topology of the router's domain and routes to networks outside of a routers domain such as routes to networks in other Autonomous Systems. (In OSPF, if areas are configured in the AS, the database will also include routes to networks in other areas.)



Each router runs an algorithm on its link-state database resulting in shortest-path tree. This shortest-path tree contains the shortest path to every router and network that the router can reach. From the shortest-path tree, the cost to the destination and the next hop to forward a datagram to is used to build the router's routing table. Distribution of LSAs and database synchronization is accomplished with the flooding algorithm.

Network types supported by OSPF

An IP router's function is to route datagrams to IP networks; the link-layer protocol (e.g., Arpanet's End-to-End protocol, X.25, Ethernet) has the job of routing a datagram to a specific host. OSPF supports point-to-point networks (a network that connects a single pair of routers, say, via a T1 link), broadcast networks such as Ethernet, and non-broadcast multi-access networks such as X.25 public data networks.

Broadcast and non-broadcast multi-access networks are similar in that they support many attached routers (but differ, of course in their ability to broadcast). OSPF uses multicasting for broadcast networks, and requires that neighbors be configured on non-broadcast multi-access networks.

Because multi-access networks support many attached routers, they can produce a great amount of routing exchange traffic on the network. To condense this traffic, every multi-access network has a *Designated Router* and *Backup Designated Router*. Each router on a network exchanges link-state information with the Designated Router which in turn generates LSAs on behalf of the network. The Backup Designated Router's job is to take the Designated Router's place in the event of its failure.

Hosts can participate in OSPF routing, but must be configured as a network with only one router on it (a stub network). We will now explore OSPF in more detail.

Adjacencies

"Adjacent" is an OSPF protocol state that two neighboring routers are in when their link-state databases have become synchronized. The flooding algorithm ensures that they continue to stay synchronized. Not all neighbors become adjacent, however. On a multi-access network, routers become adjacent with the Designated Router (and Backup Designated Router) only. A brief description of the process that leads to an adjacency follows.

Discovering OSPF neighbors

OSPF's Hello protocol initiates and sustains bidirectional relationships between neighbors. The relationship is sustained by periodically exchanging Hello packets with the neighboring router. When a router sends a Hello packet, it includes a list of all the routers from which it has recently received a Hello packet. A router confirms that it is seen by a neighboring router when it sees itself included in its neighbor's list.

On a broadcast network, Hello packets are sent using multicast; neighbors are therefore discovered dynamically. On a non-broadcast multi-access network, routers that have the potential to become Designated Router have a configured list of all routers on the network. The Hello protocol is also responsible for electing a Designated Router on multi-access networks (point-to-point networks do not have a Designated Router). The outcome of the Hello protocol is that some neighbors become adjacent. In our example, say the routers on network 3 (R1, R7 and R8), a broadcast network, have just been powered up. Each router will start multicasting Hello packets. Soon, each router will note in its Hello packet that it has recently received a Hello packets from the other routers on the network (e.g., R7 has heard from R1 and R8); each router can then be assured that bidirectional communication has been established.

continued on next page

OSPF: An internet routing protocol *(continued)*

After bidirectional relationships have been established, the Designated Router (and Backup Designated Router) will be elected (in this example we will say that R7 has been elected Designated Router).

Forming adjacencies

After bidirectional relationships have been established and the Designated Router is elected, each pair of neighboring routers decides whether or not to establish an adjacency. An adjacency will always be formed on a point-to-point link, but on multi-access networks, an adjacency is only formed between Designated Router (and Backup Designated Router) and each of its neighbors. In our example, R7 and and R1 will become adjacent and R7 and R8 will become adjacent.

When an adjacency is formed between a pair of routers, a well defined sequence of events takes place resulting in the router's databases becoming synchronized; Routers then exchange link-state information with their adjacent neighbors. The link-state information is flooded throughout the routers' domain.

The value of the Designated Router can now be more clearly seen. If each neighbor established an adjacency with every other neighbor on a multi-access network with n routers on it, there would be $O(n^2)$ adjacencies formed, which could produce a significant amount of routing traffic. With a Designated Router the synchronization problem (forming and maintaining adjacencies) is reduced to O(n).

A pair of routers are fully adjacent when their link-state databases are synchronized. It should be noted that routes are not computed using OSPF until the databases have been synchronized.

Database synchronization

The process of database synchronization is as follows. A pair of routers that are attempting to establish an adjacency send a summary of their link-state databases to each other. This summary consists of a list of abbreviated LSAs. Each router then builds a list of requests for LSAs that it needs to bring its database up to date, based on the summary received from its neighbor. A router builds this list by checking its link-state database for a copy of each LSA that has been received in the summary; if it doesn't have a particular LSA in its link-state database, or it determines that its neighbor has a more recent version of an LSA, the LSA is added to the LSA request list.

Each router will then send this LSA request list to its (soon to be adjacent) neighbor. Each router responds to the LSA request list with the LSAs that have been requested by its neighbor. The neighbors are fully adjacent when they have received all of the requested LSAs.

For example, say R3 is being powered up and establishing an adjacency with R2. R2's database summary will include its self-originated LSA and LSAs from R1, R4, and R5. R3 will request and receive all of R2's LSAs. R3's database summary includes its self-originated LSA which notes its connection to the stub net N5; R2 will request and receive this LSA. R2 and R3 now have up-to-date topological databases, and are fully adjacent.

Once the routers have established a full adjacency, they will run the SPF algorithm, add OSPF routes to their routing tables, and periodically exchange LSAs.

Maintaining an up-to-date Link-State database

Database synchronization is maintained on fully adjacent routers with the flooding procedure. OSPF has a reliable link-state flooding procedure. When an LSA is flooded it is passed from a router to an adjacent router until it has been distributed throughout the router's domain. (In this way R5, for example, will know about R2's adjacency with R3.) The decision of a router to pass on the link-state advertisement to its adjacent neighbor, however, is based on many conditions (e.g., to avoid passing on old or self-generated LSAs). The reliability is accomplished by requiring that the transfer of an LSA be acknowledged by the adjacent router; the LSA is retransmitted until it is acknowledged.

All routers keep their link-state databases synchronized by aging them and updating them if an incoming LSA is newer than the one in the database. Each LSA has a link-state sequence number field, which enables the detection of old and duplicate LSAs. Every LSA also has an age field which is used to maintain the link-state sequence space; each instantiation of an LSA has a finite life span. When a router periodically generates a new LSA, the next sequence number is used. An LSA's age is periodically incremented while residing in a router's link-state database; it is also incremented on each hop in the flooding procedure. An LSA can reach an age where it is no longer used in the flooding procedure and eventually it is flushed from the link-state database. A more common event will be for an LSA to be replaced by a more recent LSA which will contain a newer sequence number. Whenever it is determined that there is a change in the topology, the SPF algorithm is rerun.

OSPF Area Routing

OSPF permits contiguous networks and hosts to be grouped together. This group, along with the routers that have interfaces connected to the networks in this group, is called an *area*. (Constructs such as areas are configured into the AS.) All routers within an area keep and maintain an identical topological database. Routers that are in more than one area (i.e., the router has two interfaces, each connecting to different networks that have been configured to be in different areas, aka *area border routers*) keep a distinct topological database for each area that they are in and therefore run the SPF algorithm on each area's database (for example R6 keeps a topological database for Area 1, Area 2 and the backbone area). An area is a generalization of a subnetted network; it should be noted that all subnets of a network must be contained within a single area.

The area construct has the following advantages: An area greatly reduces the routing exchange traffic within an AS. One example of this is that all subnetting information is condensed when passed outside of the area; routers external to an area treat subnets within an area as if they were a single network.

When hosts and networks are grouped into an area, the area's information is hidden from routers outside of the area, and routers within an area are protected from information external to the area; in other words, configuring an area builds a fire wall.

It should be noted that for some routing protocols, the definition of an area includes the notion that the area ID is part of the destination address. Since OSPF is designed for the IP environment, OSPF's notion of area allows routing to a specific network within an area or a router that has an interface to another AS within an area.

continued on next page

OSPF: An internet routing protocol *(continued)*

Another point to note about OSPF area functionality is that OSPF does not attempt to repair area partitions. When an area is partitioned, each partition will become a separate area. If the partitions are connected via the backbone (described below), there will be no loss of reachability between the partitioned areas. Repairing partitions is complex; OSPF has no partition repair worries.

The OSPF backbone

Routing within an area (intra-area routing) is the lower level of the OSPF routing hierarchy. Areas are connected to each other via the spine of the AS called the backbone. In other words, OSPF routing can happen at two general levels: the lower level routing within a group of contiguous networks and hosts known as an area, and the higher level of routing that happens between areas, traversing the backbone (inter-area routing). The backbone is made up of routers that belong to multiple areas (area border routers), networks (and their attached routers) and routers that are not contained in any area. In our example, R1, R2, R3, R4, R5, and R6 make up the backbone.

It should be understood that OSFP can be configured to run flat or with an area hierarchy. The good news is that the backbone is an area. The backbone runs the same procedures and algorithms to maintain routing information within the backbone that any area would with the additional duty of distributing routing information between areas.

We have noted above that areas must be contiguous and that OSPF does not attempt to repair partitions, however, there could be an area which interrupts the contiguity of the backbone. This could be caused by router failures or possibly the AS has been configured in such a way that the backbone is not contiguous. For these reasons, the backbone can be configured to have an additional type of connection to a neighbor called a *virtual link*.

A virtual link is part of the backbone and can be configured between any two area border routers as long as the intervening area is the same area (but not the backbone). Virtual links are handled by OSPF much like to point-to-point links. In our example, R6 is an area border router for Area 1 and Area 2 but R6 is isolated from the rest of the backbone; there would therefore have to be a virtual link configured between R1 and R6.

As an example of intra-area and inter-area routing, say a datagram is sent by a host on N2 in Area 1 that is destined for a network in Area 3. It is first routed within Area 1 to an area border router in Area 1 such as R1. It is then routed across the backbone to Area 3's area border router (R4), and finally routed to the destination router in Area 3. With this scheme the shortest path is chosen at each step of the way; in Area 1 from the source to an area border router, in the backbone from Area 1's area border router to an area border router for Area 3, and in Area 3 from the area border router to the destination.

External routes

The highest level of the OSPF hierarchy is inter-AS routing. An AS border router (i.e., R2 in our example) may learn about these external routes from another routing protocol such as EGP, or through configuration information. External route information is flooded throughout the entire AS.

Conclusion

The functionality that is provided by OSPF is long overdue. OSPF makes IP-based SPF routing technologies openly available and leaves the needed room for continued internet growth. OSPF eliminates many complexities and problems that have arisen from exceeding the limits of the current open IGPs. With OSPF, an AS administrator can base the configuration of the routing topology of a large AS on the limits and the load requirements of the AS. In general, OSPF provides a routing context within which aspects of internet routing, such as type of service based routing, variable length subnet masks, IP multicast, and area routing, can be explored.

There are currently two working implementations of OSFP; one done by John Moy, and one done on a UNIX platform at the University of Maryland. The OSPF internet draft

DRAFT-IETF-OSPF-SPEC-00.PSZ.

is available by anonymous FTP from SRI-NIC.ARPA in the INTERNET-DRAFTS: directory. [Ed.: See page 52 for more information about the INTERNET-DRAFTS: directory]

ROB COLTUN is a Systems Programmer in the Network Infrastructures group at the University of Maryland where he is currently working on a UNIX-based OSPF prototype. Rob previously was a member of the technical staff at the MITRE Corporation where he worked on an ISO VTP forms mode implementation, MITRE's Exterior Gateway project, and MITRE's mini-TAC prototype. Earlier, Rob worked for BBN.

Routing Glossary

Like most technical fields, routing has its own language. Here are a few of the most frequently used terms which you will find references to in this special issue.

Count-to-Infinity: A problem that occurs in distance-vector style routing algorithms (also known as Bellman-Ford, Old Arpanet, etc.) where routers mis-interpret old routing information as new, thus causing loops. Simplistic distance-vector algorithms would only purge old routing information after the distance metric would increment itself to some number larger than the longest possible path, thus the name count-to-infinity.

Hold-Down: A solution to the count-to-infinity problem where routers refuse to use new information for a period of time called the *hold-down time.* If the hold-down time is longer than the time it takes for the distance-vector routing algorithm to count to infinity, then loops are avoided.

Split-Horizon: A partial solution to the count-to-infinity problem. It only solves the problem for loops between two nodes (sometimes called a *ping-pong loop*). In split-horizon, a router only uses another router for its next hop if the other router isn't already using it.

The IETF Open Routing Working Group

by Marianne Lepp, BBN Communications Corp.

Introduction

The Open Routing Working Group (ORWG) of the Internet Engineering Task Force (IETF) is chartered with solving the problem of routing between *Autonomous Regions* (ARs) with policy constraints. Over the past year ORWG has clarified the problem it is facing and designed an architecture to solve it. The architecture is a source-based link-state protocol with route set-up and a conceptual framework that permits data reduction and hence will function in a very large Internet. Our design routes on the basis of Autonomous Regions. That is, inter-Autonomous Region routing will route from one Autonomous Region to the next. Internal AR routing is kept separate and independent of inter-AR routing. (Open Routing refers to the inter-AR routing, while *Internal Gateway Protocols* (IGPs) provide the intra-AR routing.) Paths to end systems will be computed by the AR that contains them.

This article will discuss our topology and policy requirements. A forthcoming RFC will discuss the architecture.

The working group is part of the Internet Engineering Task Force (IETF). We have been working closely with the Autonomous Networks Task Force (ANTF) which has collected policy requirements and written a policy description language. We are also working with the Privacy Task Force (PVT) which is working on some of the authentication and validation requirements of the protocol.

The members of the ORWG are Marianne Lepp, BBNCC, chair; Ross Callon, DEC; Noel Chiappa, Proteon; Dave Clark, MIT; Pat Clark, Ford Aerospace; Deborah Estrin, USC; Robert Hinden, BBNCC; Michael Little, SAIC; Tassos Nakassis, NIST; Zaw-Sing Su, SRI; Paul Tsuchiya, Mitre; and Lixia Zhang, MIT.

Documents

Our first activity was to define our requirements. This document was published as IDEA 007, *Requirements for Inter-Autonomous System Routing* (Jan. 1988, R. Callon ed.). We are currently preparing four additional documents. *Functional Requirements for Inter-Autonomous System Routing,* an updated statement of requirements, outlines the functionality required of the routing architecture for the Internet. The *Architecture of Inter-Autonomous System Routing* will present the basic ideas of the protocol. Both should be available as RFCs by the end of summer 1989. The architecture RFC builds on RFC 1102, *Policy Routing in Internet Protocols* (Jan. 1989, D. Clark). The third document is a functional specification, which describes the functionality provided by this protocol. This document will be available this fall. We expect to have the protocol specified for prototyping written by the end of 1989. Formal publication will be later because we expect more feedback and discussion to match the greater detail of a specification. It is important to note that the full routing architecture comprises a number of different protocols, not all of which will be completely specified by the document we prepare.

In this article we use the term *Autonomous Region* (AR) to mean an administrative entity with a coherent set of policies, which, therefore, presents a uniform policy and administrative interface to the Internet. For the purposes of this paper, an AR can be thought of as either an Autonomous System or an ISO Administrative Domain.

In this article we present an overview of why policy is important, discuss the kinds of policies we intend to support, and discuss our model of the Internet in which this protocol will run.

The term *policy* has been used so diversely it is worth spending some time here to describe what we mean by policy. We define the term policy to mean an administratively determined restriction on the flow of data. In particular, we use policy to refer only to administrative issues and do not use it to refer to mechanisms such as the selection of routes by hand-collated tables. We also distinguish policy from *type-of-service*. The latter refers to physical attributes of a path, such as bandwidth or delay. Both policies and type-of-service bear on routing in that the routing protocol must select routes that honor policy and type-of-service requirements.

Policy

Policies arise for many reasons, including: increasingly diverse funding sources; a push for users to pay for their traffic; transmission media providers who have recognized the financial potential of packet-switching, and are likely to provide more widely available packet-switching services as well as more appropriate and varied tariffing of these services; and the pervasiveness of data networking in the workplace along with attendant requirements for privacy.

When only one agency, the Department of Defense, funded all the networks that comprised the Internet, there were no explicit policies. Networking research was (and still is) viewed as a national priority; to support it, network attachment was funded by the government. Access was available to all research sites willing to pay a highly subsidized connection fee. As the technology has matured in the past decade or so, packet-switching has moved into the private sector. It is no longer viewed as requiring the same level of federal underwriting, and network providers have grown increasingly unwilling to provide services to arbitrary users.

At the same time, large Internet components have been established to meet the needs of a wider research community and a broader academic base. These components are under the sponsorship of agencies other than the DoD, e.g., the National Science Foundation, the Department of Energy, and NASA. Each agency envisions a mission for its networks that often restricts the use of its resources. Each agency must secure its own funding and thus has no incentive to provide universal transit service for other agencies' users.

Policies can be classified as *user policies* or *provider policies*. A user is an application process or the source AR. A provider is whatever transit systems and backbones the data path traverses. A user policy is determined unilaterally by the source user, and applies to the entire path. A provider policy is one that is determined by a provider. The routing algorithms must synthesize the provider policies for each AR the path crosses.

Examples

A few examples of each policy type will help clarify the distinction. An example of a user policy is a ceiling on the acceptable cost of the path. Another example is the restriction that the data transit only secure systems. In this case, the user might want not only to control the transit systems in its path, but also to hide the fact that such control is being exerted.

continued on next page

The IETF Open Routing Working Group *(continued)*

For example, two companies working jointly on a development project may not want the partnership made public, but announcing the agreement to use each others' systems conflicts with that requirement.

An example of a provider policy is the refusal to forward traffic for non-paying customers. Other provider policies include restricting the use of resources to certain ARs (e.g., ARs that have shared in the cost of establishing the transit resources), or to certain classes of ARs (e.g., government systems), or to certain classes of users (e.g., university researchers). There can be restrictions based on the high level protocol in use (e.g., Mail, FTP, Telnet) or the time of day. ARs that are providers of such restricted transit services express the provider policies for enforcing such restrictions.

It is obvious that routing cannot accommodate every conceivable policy, but a fairly extensive set of them can be met, and a mechanism for manually specified routes will accommodate the rest. The Autonomous Networks Task Force, chaired by Deborah Estrin, has been collecting data and writing requirements that identify the most important provider policies (*Requirements for Policy Based Routing in the Research Internet,* D. Estrin, preprint).

Evolution of the Internet

The Internet currently comprises over 850 networks. The number of networks has been doubling in size every year for the past several years. It is not known if this rate will continue, but the routing architecture must be designed for that possibility.

As the Internet grows, problems caused by complexity and size arise. The problems of concern to us include 1) costs—the cost of the routing computation and distributing the result of that computation, the cost of maintaining a current view of the Internet topology, and the cost of forwarding packets, 2) the complexity of the policy specification and of the subtlety of serendipitous policy interactions, and 3) the size of the policy and routing databases. The relative importance of each of these issues depends on the precise way the Internet grows.

The topological complexity of the Internet has increased along with its size. In 1983 the Internet consisted of a single core system providing inter-connectivity to networks joined to it in a tree-like structure. The Internet as it now exists is something less than a full mesh and quite a bit more than a tree. It is basically a three-level system with the NSFnet and DDN providing backbone service to Regional components which in turn provide connectivity for campus-level systems. The strict hierarchy is subverted by connections across levels and by-pass connections from lower levels directly to the NSFnet or DDN backbone.

The Internet will probably continue to evolve in its current three-tier pattern, but that evolution could increase the strictness of hierarchy or continue to increase the complexity of the mesh. Many of the factors that led to the current form of the Internet are likely to remain important and to result in exceptions to any strict hierarchy: the need for redundancy, the diversity of the funding sources, specialized requirements for certain user classes, the absence of a central authority, and the absence of acceptable alternatives.

As long as the Internet is composed of interacting systems which are diversely funded, policy, in the form of explicit restrictions on the flow of data or in the form of tariffs, will be important. As long as different classes of users have widely differing needs, private ARs will come into being, and links that cut across hierarchy levels will exist. As long as redundancy is a key requirement, the Internet will gravitate away from a strict hierarchy. As long as users require autonomy and independence, they will build their own backbone networks. Thus, we do not see the Internet naturally evolving into a simple system based on direct connections to a single ubiquitous backbone. At the same time, we look forward to the Internet making an increasing use of common carriers and a more rational topological structure.

The pull toward a stricter hierarchy or toward a more full mesh topology determine the complexity that the routing protocol will face. For example, some factors that bear on complexity are: the number of cross-links and links that cut across level boundaries, the visibility of these links to routing, the depth of the hierarchy, the length of typical paths, the geographic and topological diameter of the Internet, whether policy can be formulated so that ARs provide a useful clustering mechanism, the proportion of the ARs that are transit systems, limited transit systems, or stubs, and the degree of cooperation between systems.

Based on these observations, we propose a model for Internet growth. We believe that this is an accurate vision of the future. We are, nevertheless, designing a routing protocol that will be robust in the face of different patterns of Internet evolution.

Working Model of Internet Growth

We believe that the Internet will continue to develop in its current patterns. We anticipate relatively few backbone ARs will provide a variety of services ranging from toll access for large numbers of users to very high-speed (gigabit and beyond) transport for a small set of users. Most likely there will be connections between these backbones. There will be local sites equivalent to academic campus networks and campus-level ARs. These will be connected to some set of mid-level ARs which in turn will be directly connected to some subset of backbones. Add to this simple picture two tiers of mid-level systems, connections between local sites, and connections from local sites directly to a backbone. This is our model of the Internet in the next five years. (See Figure 1).

Our design plans for millions of local networks (most of which will be invisible outside their Autonomous Region), and tens of thousands of ARs visible to routing.

Stub systems

Most of these tens of thousands of ARs will be *stub* systems, systems that do not provide transit service for any other ARs. Some, perhaps many, of theses ARs will provide a very limited amount of transit service, determined by bi-lateral agreements between AR administrations. This service will be carried on semi-private links whose existence need not be generally broadcast. Still other ARs will provide transit service to a wide range of sources and destinations. These are the backbones.

continued on next page

The IETF Open Routing Working Group *(continued)*

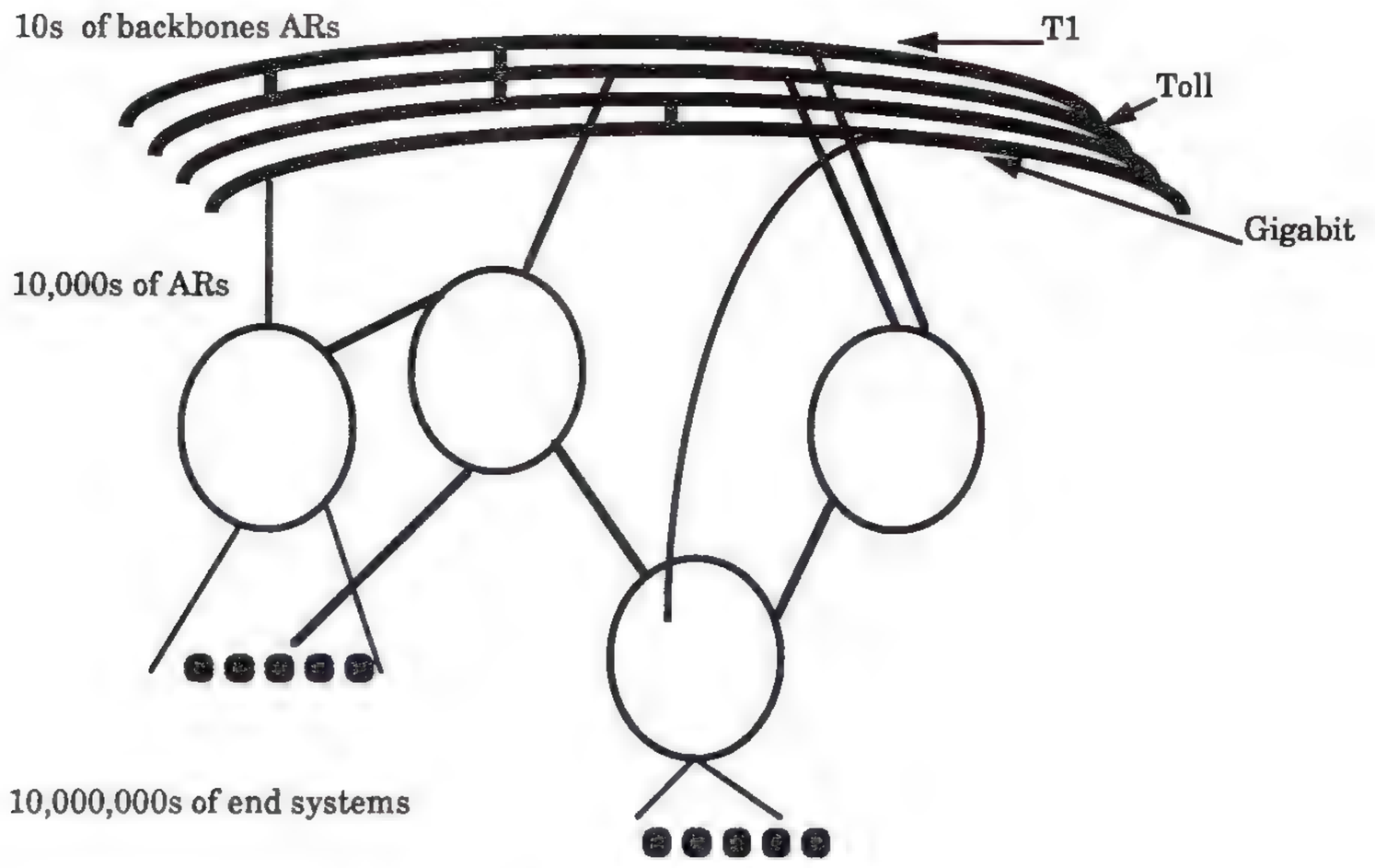


Figure 1: Future of the Internet

As the need for Internet connectivity grows there will be dual pressures on the backbones. Policies have, by their very nature, a local or myopic view of the world which is orthogonal to universal connectivity. ARs will continue to negotiate direct links to their favored destinations (the source of pressure for cross-links and short-cuts). On the other hand, they will need to find good connections to a set of backbones for universal connectivity. The requirement for this connectivity is the incentive for vendors to provide transit service to customers, pushing toward a simpler, more hierarchical structure for the Internet.

The requirements that this imposes on routing are that routing must handle mesh-like connections, but that efficiencies afforded by the three- or four-level system can be incorporated into the protocol. It is a requirement that the connections that do not follow a hierarchical pattern must be accommodated.

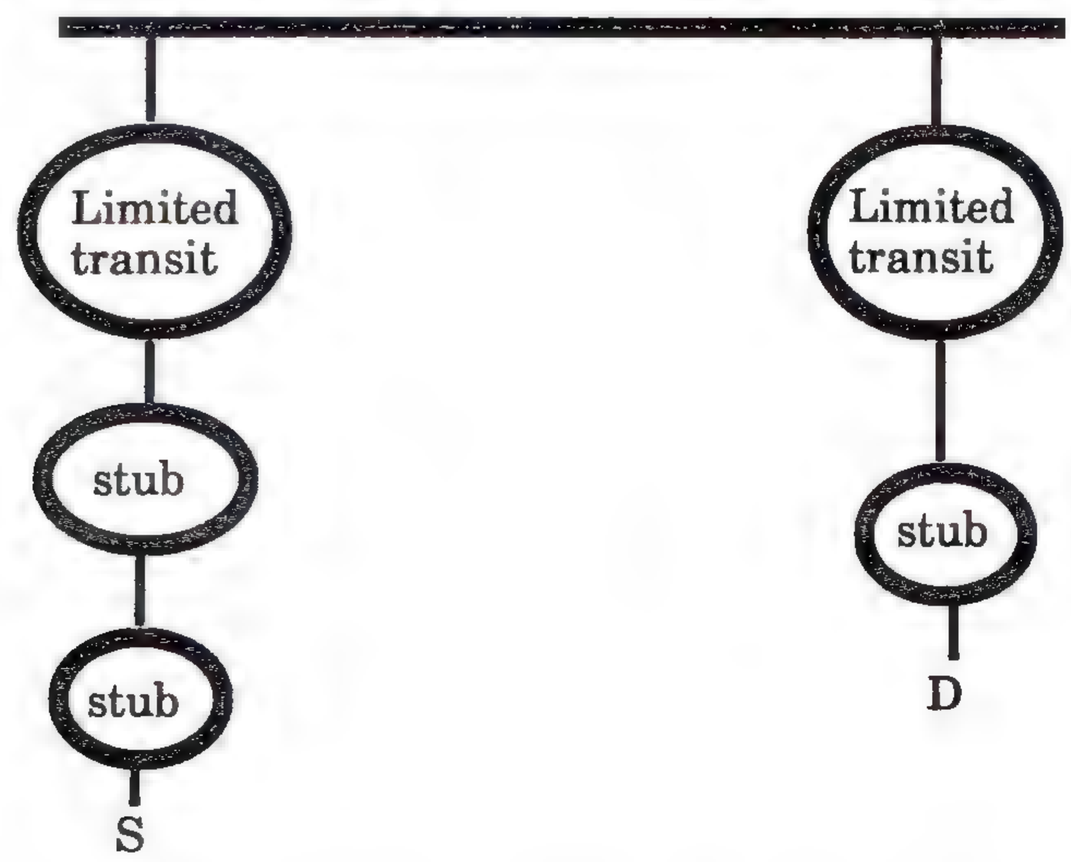


Figure 2: Typical Path in the Internet

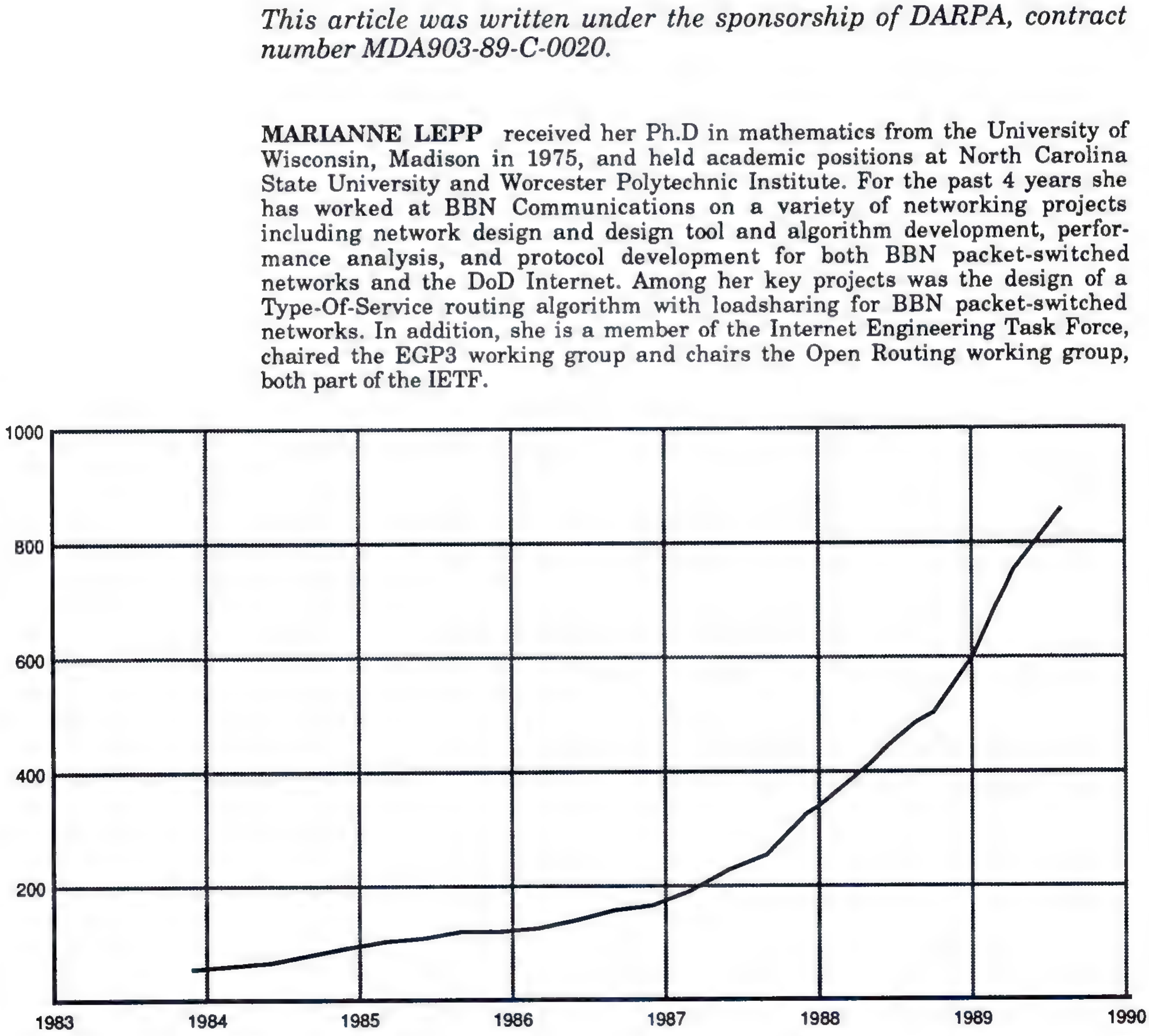
A number of design efficiencies are possible if we build the dynamic aspects of routing at the AR or bundle of AR level. We believe that the number of such systems that accept transit traffic will be relatively small. We can take advantage of the expectation that the length of paths will tend to be on the order of 5 or 6 AR hops rather than 200. In addition, if we anticipate millions of end systems subsumed under administrative wings that provide paths from the AR entrance gateway to these end systems, inter-AR routing is relieved of the need to know how to route to millions of entities (but left with the need to find their addresses.)

Conclusion

The Open Routing Working Group is designing a protocol to route data between Autonomous Regions. This protocol will accommodate policy restrictions on the flow of data and can operate in a range of Internet topologies: in one that has the form of the current Internet or in an Internet that has a more hierarchical or more mesh-like structure. In designing the protocol we are cognizant of the issues of Internet size and complexity. The architecture itself will be described in a forthcoming RFC.

This article was written under the sponsorship of DARPA, contract number MDA903-89-C-0020.

MARIANNE LEPP received her Ph.D in mathematics from the University of Wisconsin, Madison in 1975, and held academic positions at North Carolina State University and Worcester Polytechnic Institute. For the past 4 years she has worked at BBN Communications on a variety of networking projects including network design and design tool and algorithm development, performance analysis, and protocol development for both BBN packet-switched networks and the DoD Internet. Among her key projects was the design of a Type-Of-Service routing algorithm with loadsharing for BBN packet-switched networks. In addition, she is a member of the Internet Engineering Task Force, chaired the EGP3 working group and chairs the Open Routing working group, both part of the IETF.



The Internet continues to grow at a rapid rate. There are now over 850 connected networks.
(Graph courtesy of Steve Atlas, BBN Communications.)

Adopting a Gateway

by Bob Enger, CONTEL Federal Systems

Introduction

The Adopt-A-Gateway (AAG) program was an informal attempt to address the problem of congestion in the Internet core gateway system, specifically the mailbridge and EGP server gateways. Rather than attempting to deal with the tough problem of providing a genuine, long term solution, this program simply applied a band-aid to the situation. It provided some immediate relief; buying time to work on the long term solutions. This article provides a brief review of the AAG program, a cursory description of a couple of the congestion culprits, and a summary of the current status. Hopefully, by the time of publication, enough long term solutions will have been implemented that the topics discussed here will seem like a stroll down memory lane.

(Re-)discovery

When CONTEL Federal Systems became part of the Internet, we often noticed that performance was much worse from hosts connected to our local network, than from our gateway connected directly to Net 10 (Arpanet). Experimentation with the Sun *vcstat* program showed that under certain circumstances the core gateway system was providing genuinely different packet routing when traffic originated from hosts on our local network. We had rediscovered the infamous *extra-hop problem*, one of the culprits causing avoidable congestion.

Background

An internet is formed when gateways are used to inter-connect multiple (sub)networks. Traffic from a host on one network can be routed to a host on an adjacent, or non-adjacent network by traversing some set of intervening gateways. Arpanet and MILNET have come to act as an internet backbone, inter-connecting the gateways of many other (sub)networks.

Arpanet and MILNET are in turn connected by a group of gateways referred to as *mailbridge* gateways (out of the fear that some day they will be restricted to only routing packets bearing mail-protocol traffic). The mailbridge gateways exchange their routing information by using the *Gateway to Gateway Protocol* (GGP). Certain other networks also connect to Arpanet or MILNET using GGP speaking gateways. Collectively these are known as the *core gateway system*. GGP implements a shortest-path type routing system amongst the core gateways, allowing them to be connected and to route in an arbitrary fashion. GGP routing updates contain only the distance to each network. They do *not* indicate what the address of the next hop is, and that causes problems.

Since GGP is proprietary to BBN, and not all networks attach to the core using BBN gateways, some method of allowing non-core gateways to advertise themselves must be employed: enter the *Exterior Gateway Protocol* (EGP). EGP provides only reachability information. The distance values in the routing updates from various sites are not uniformly administered, and therefore cannot be trusted to provide loop-free, much less shortest-path, routing. For this reason, the Internet utilizes EGP only for its ability to indicate that a gateway is reachable and to what networks it provides access. EGP must not advertise networks that could form an internet topology susceptible to routing loops. Thus, EGP can only be trusted to provide routing in a loop-free, branching tree organization.

To make these two camps of gateways inter-operate, and to cut down on the routing-protocol traffic, three core gateways on both Arpanet and MILNET have been equipped to speak EGP (as well as their native GGP). These machines have come to be referred to as the *EGP-core gateways*. They serve two functions: to import EGP advertised reachability information into the GGP speaking camp, and secondly to re-broadcast the EGP (and GGP) reachability information back to the EGP-speaking camp. The EGP-core machines do not speak EGP amongst themselves. They learn of each others routes via the same GGP routing updates used to inform the rest of the core system.

Extra-Hop

Hopefully by now, you're asking yourself, what is "extra hop?" Well, extra-hop comes in two varieties, EGP and GGP.

EGP extra-hop can occur between two non-core gateways when they do not peer with a common EGP core server. When each of these non-core gateways receives its routing update from its peer EGP core server, it does not contain a direct routing to the other non-core gateway's nets. This happens because neither gateway was peered with the EGP core server providing the update to the other. Instead, each of these EGP core servers sends its non-core peer a routing update saying, "to reach the nets behind the other non-core gateway, send your traffic to the other EGP core server!"

The EGP core servers have direct knowledge of only those non-core gateways that are their own EGP peers. The EGP core servers share their EGP learned reachabilities amongst themselves using GGP.

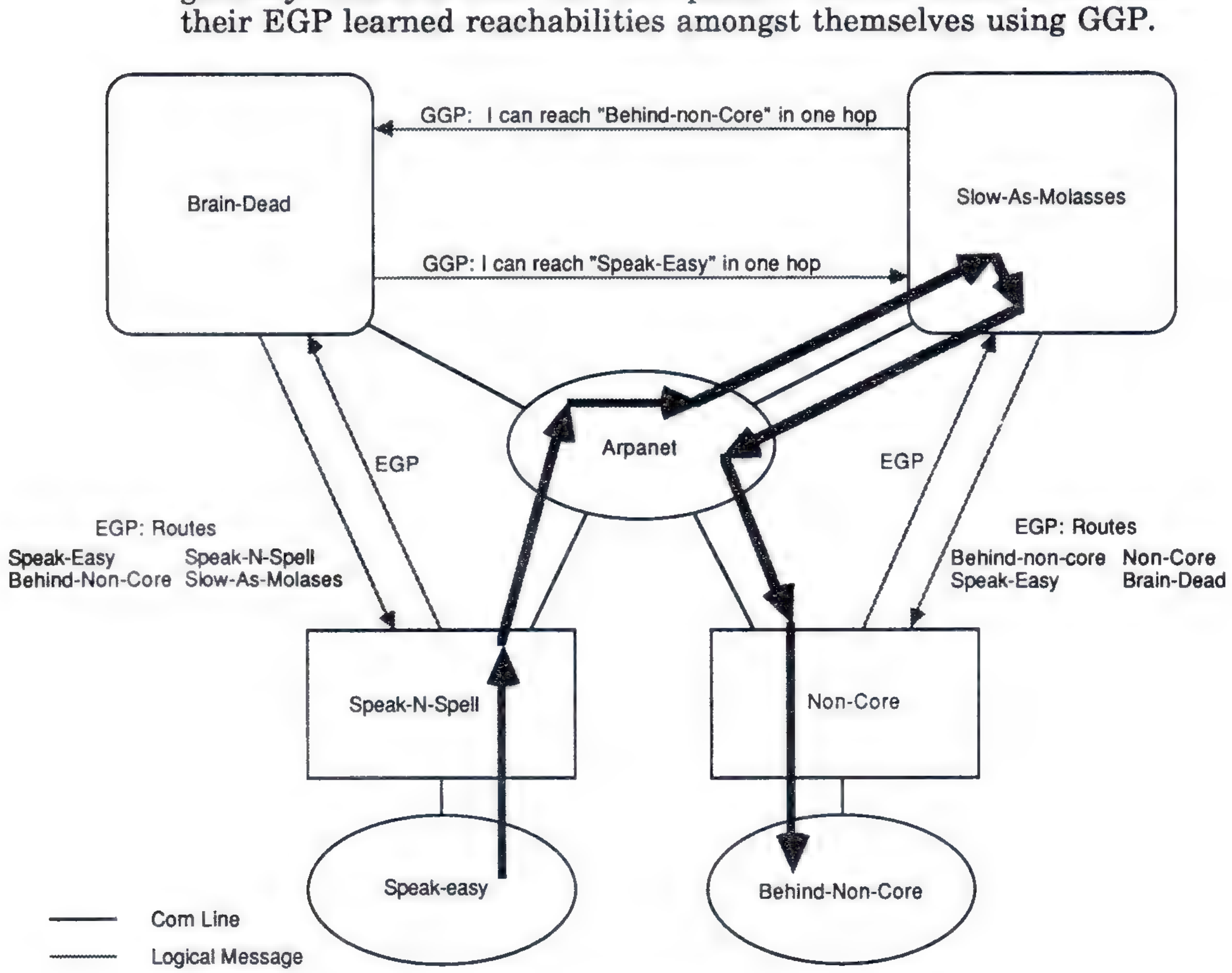


Figure 1: EGP Extra-hop example

continued on next page

Adopting a Gateway *(continued)*

Referring to figure 1, gateway Slow-As-Molasses, the EGP core server peered with gateway Non-Core sends out GGP routes like:

> "Hi, I'm gateway Slow-As-Molasses. I can reach network Behind-Non-Core in one hop."

Unless it is also peering with gateway Non-Core, another EGP-core gateway will have only this GGP propagated route to network Behind-Non-Core, and will distribute this route to its own EGP peers. Thus, Brain-Dead's peers think their only way to get to network Behind-Non-Core is by sending traffic to Slow-As-Molasses, which lives up to its name. A way to preclude all of this is to insure that every combination of source and destination non-core gateway has at least one EGP-core gateway peer in common.

GGP extra-hop arises from the same problem as EGP extra hop: GGP does not communicate who the next gateway really is. All GGP says is, "I can get to your destination net in X-hops, trust me." A core gateway that obtains its routing data solely via GGP will not "see" any non-core gateways. Indeed, the only info it will receive about networks behind non-core gateways will be from the EGP-core servers. Since GGP does not allow the EGP-core server to convey what gateway is the next hop after itself, the only information it ends up sending to its GGP friends is: "I can reach network X in Y-hops."

Figure 2 shows the routing information exchanged in a theoretical piece of the internet. Core gateway DOA is a non-EGP speaking "mailbridge" gateway. Brain-Dead and Catatonic are EGP-core servers. Each has a non-core gateway as an EGP peer; Speak-N-Spell, and Stealth, respectively.

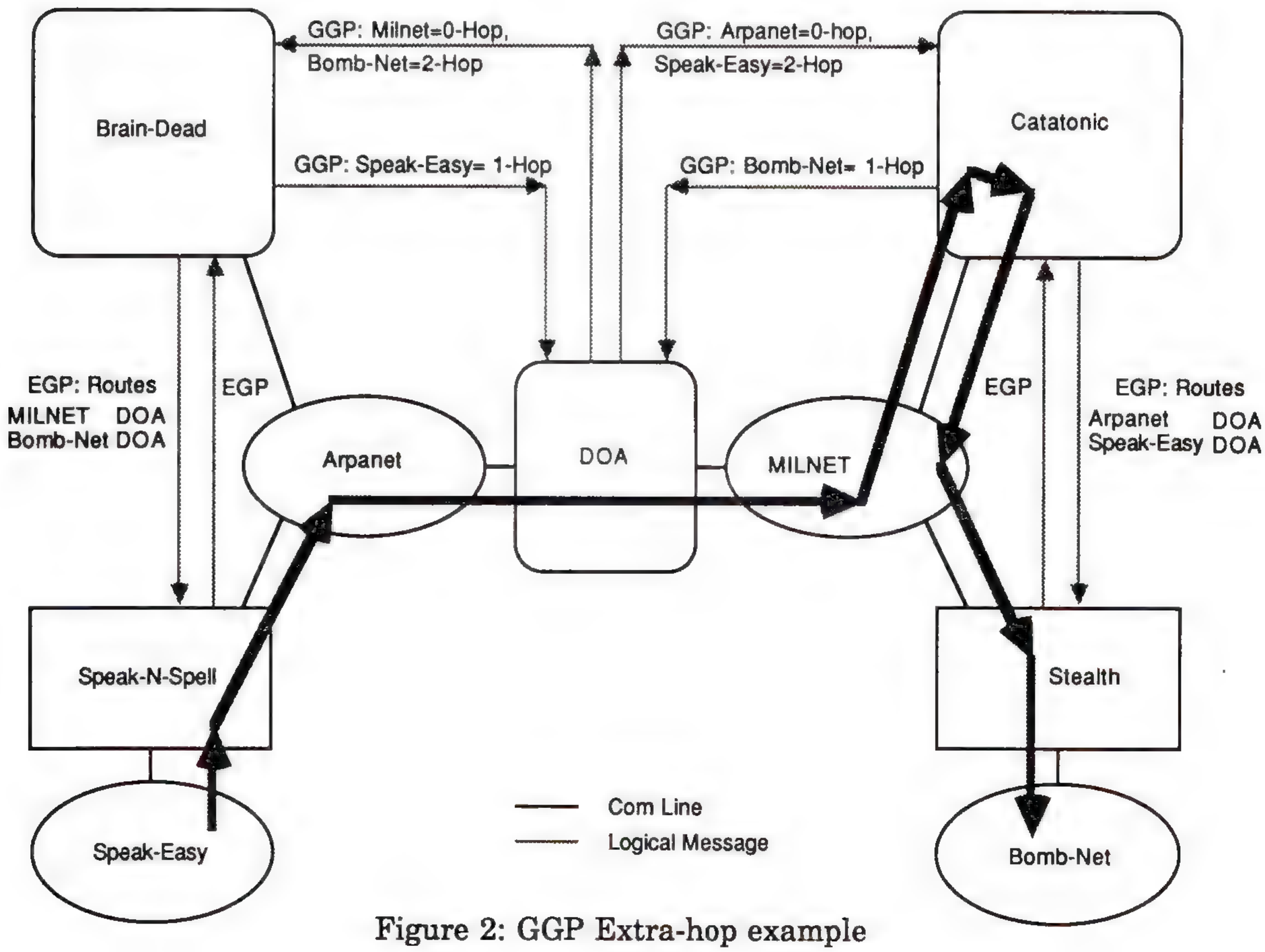


Figure 2: GGP Extra-hop example

The dotted arrows show the logical flow of routing information. Note that although DOA is "directly" connected to Stealth (via MILNET), the only route to Bomb-Net seen by DOA is the one from Catatonic. Likewise, DOA thinks Brain-Dead is the best next-hop gateway leading towards network Speak-Easy.

The figure illustrates the data flow from Speak-Easy to Bomb-Net. The data flow's useless excursion through Catatonic gives rise to the phrase "extra-hop." Similarly, when traffic flows from Bomb-Net towards Speak-Easy, it will make an extra-hop through Brain-Dead.

Both EGP and GGP extra-hop load the EGP core servers with extraneous through-traffic. Since these gateways have about as much processing horsepower as a digital wristwatch, they become quite overloaded! Additionally, the double traversal of the (sub) networks increases the load on the networks.

Have a slooowww day!

When traffic made a hop directly across the Arpanet, or the MILNET, delay times were tolerable. However, when traffic was routed through one of these powerhouse core gateways, delays went through the roof, and packets started to be dropped. Dropping packets is especially bad because the retransmissions rob the (sub)networks of bandwidth that could go to new traffic. A good audience (your cue) should now be asking, "what should be done to alleviate this situation?"

Stuck with GGP

Well, the obvious solution is to replace GGP with a *smarter protocol*. However, it was felt that it would be a waste of dollars to implement GGP's successor in the same relay and plug-board logic. DCA made a decision that the GGP replacement should be coupled with the retirement of the LSI-11 gateways (the Smithsonian Institution had been hounding DCA to contribute them for a dinosaur exhibit anyway). Thus, the replacement of GGP became entangled with, and bogged-down by, the Butterflys' slow passage through the funding swamp. I've read mail dated November 1986 saying that the Butterfly gateways would soon save the day; apparently they spend a long time in the cocoon. The mailbridge and EGP-core gateways are still running.

Brain transplants

Given that we were stuck with GGP, and user traffic was routing through hardware waiting to be taxidermied, I wondered what could be done to provide some immediate relief? How about a brain transplant? It worked for Dr. Frankenstein! Maybe we could raise the intelligence of an LSI-11 gateway up from the level of a rock, to something akin to a trilobite. Thus, even if the packets continued to needlessly flow through the EGP core servers, at least there would be more CPU power to handle the load; round trip delay times wouldn't approximate our lunch hour.

The idea was put forth at a meeting of the Internet Engineering Task Force (IETF) and initially received little general support. However, Mike Karels of U.C. Berkeley apparently saw merit in the cranial replacement concept. He rounded up the right people after the session and got them talking. In all likelihood, were it not for Mike's intervention we would be living with LSI-11/23s to this day.

continued on next page

Adopting a Gateway *(continued)*

The first patients

The group Karels assembled at the IETF meeting included Mike Brescia of BBN, and Mike Petry and Louis Mamakos of the Univeristy of Maryland. Mike Petry and Louis said they had access to 11/73 CPU cards and would see to the upgrade of some of the Arpanet EGP-core gateways. Mike Brescia suggested that I contact BBN's LSI-11 gateway curator, Steve Atlas.

Within a week of the IETF meeting, Steve had located an 11/73 CPU somewhere inside BBN and had upgraded the Arpanet EGP-core gateway located at BBN. Mike Petry and Louis had upgraded the remaining two servers by November 23

With the new brains in place I repeated some *ping* tests against a host connected directly to MILNET. When performed from hosts connected to my local network, the returning traffic was subject to GGP extra hop. The delay times had dropped by one to two orders of magnitude. Performing the same test from my Net 10 connected gateway revealed lesser, but still substantial delay, with large variance, and sometimes high packet loss. Since both Arpanet and MILNET are reliable (sub)networks, there weren't many places left to point the finger regarding packet loss. I decided to try to give the mailbridge gateways a new lease on life.

CONTEL adopts a mailbridge

I did some grave robbing inside CONTEL and located an 11/73 brain and a megabyte of memory. After a few more unholy acts I got the company's permission to "loan" the boards to the government. Since our mailbridge homing was to DCEC it seemed the perfect adoptee for CONTEL. Annette Bauman of DCA arranged an early-morning visit to DCEC wherein we upgraded DCEC-MILNET-GW to an 11/73 with full memory. We woke Steve Atlas at home and he verified our handiwork using remote diagnostics.

More Foster Parents sought

Encouraged by the performance improvements, Phill Gross and I sent a humorous mailing to the TCP/IP mailing list seeking Foster Parents for the remaining mailbridge gateways, and the EGP core gateways on MILNET. Paul Pomes and Bill Nesheim became foster parents soon thereafter.

Flashing lights

Having never seen the ACC 1822 interface cards before, I spent some time watching them while at DCEC. I noticed that what appeared to be status lights occasionally went out for periods of time. After some investigation I found out that these indicators were revealing, in effect, blocking in one direction or the other on the 1822 link. Steve Atlas informed me that the restricted memory in the mailbridges had forced them to allow only two buffers to be queued for DMA transmit and receive for each interface. With the slow CPUs this had allowed the 1822 interfaces to go idle periodically. I was seeing this even with the higher speed CPU. Steve put the new memory to good use by increasing the DMA limits.

Taking advantage of PSN 7.0

The Arpanet used to impose a form of congestion control requiring the monitoring of *Ready-For-Next-Message* (RFNM) "acknowledgements" from the PSN. A host or gateway was not allowed to have more than 8 messages outstanding (unacknowledged at the (sub) network level) for any destination. There are a variety of reasons why destinations don't acknowledge messages promptly. It may be that the destination is somehow speed limited.

Another case occurs when multiple sources saturate a destination's interface. It is easy to imagine this happening with multiple mailbridges extra-hopping to three EGP core gateways. Mailbridge gateways were good network citizens and played the RFNM counting game. When a mailbridge routed a packet it had received, it checked to see if it had too many messages outstanding for the determined destination. If it did, the packet was dropped.

To degrade matters, the mailbridge code counted a message as in-flight (unacknowledged) when it was placed into the transmit queue, not when it actually crossed the interface to the PSN.

PSN 7.0 removes the RFNM counting requirement. It allows the PSN to block clients at the link level when they have exhausted their buffer quota. Steve Atlas took advantage of this by turning off RFNM counting in the mailbridges in an attempt to reduce their packet drop rate. On the Arpanet side of a mailbridge, congestion to even a single destination may now cause the PSN to block all traffic from the mailbridge. Hopefully this doesn't happen too frequently. Since MILNET was still using PSN 6.0, one must wonder how often a mailbridge was blocked on the MILNET side when it exceeded the eight-in-flight rule. The removal of RFNM counting may also increase the chances that an unsocialized sender, or a popular destination will monopolize a mailbridge's buffer pool.

A Foster Parent with a BIG heart

I attended the IETF meeting in San Diego in March 1988, and made a short presentation on the results of the Adopt-A-Gateway program. At the end I made another pitch for Foster Parents. Phill Karn of Bellcore came forward this time. He donated *seven* 11/73 CPUs! This allowed us to upgrade all of the remaining mental-midgets.

Current status

Butterfly gateways have been been deployed as replacements for the LSI-11 mailbridges. Some mailbridge traffic has been "re-homed" to the Butterflies. The Butterfly's EGP implementation is being tested currently. A problem in the design of the event scheduler was discovered. EGP peers were constantly being acquired and dropped, leading to rapid changes in the contents of the routing updates. A patch has been developed and is being distributed. We may finally be nearing the time when the LSI's are switched off for good.

Summary

The Adopt-A-Gateway effort bought the Internet some time, and gave some tired old gateways a last fling before they hit the scrap heap. The program itself also proved that its possible for the Internet community to help themselves. Bureaucracy can be circumvented; benefit provided without profit. I would like to express my thanks to all those who gave their time and brains to the program.

ROBERT ENGER received his B.Sc in Electrical Engineering from The University of Maryland in 1981. After graduating, Bob worked at Linkabit Corporation's Eastern Operations. He wrote the firmware for the Naval Research Laboratory's satellite system Range and Synchronizer Test Unit. Thereafter, Bob became the supervisor of telecommunications for Eastern Operations. Bob then went on to join CONTEL Federal Systems where he is currently a Senior Staff Engineer. Bob supports the operation and design of the in-house computer networks, and coordinated CONTEL's connection to the Internet. Bob is a member of the IETF's User Services working group.

Components of OSI: Routing (An Overview)

by Paul Tsuchiya, MITRE Corporation

Introduction

There are currently several network layer routing standards developed or being developed in ISO. They are:

- ISO 9542: "End System to Intermediate System Routing Information Exchange Protocol for use in Conjunction with the Protocol for the Provision of the Connectionless-mode Network Service" (ISO 8473).
- ISO DP 10030: "End System to Intermediate System Routing Information Exchange Protocol for Use in Conjunction with ISO 8878."
- (No ISO number): "An IS-IS Intra-domain Routing Information Exchange Protocol."
- (No ISO number): "An IS-IS Inter-domain Routing Information Exchange Protocol."

These four standards more-or-less reflect an intentional partitioning of functions for network layer routing in ISO, as described in ISO TR 9575: OSI Routing Framework. (I say more-or-less because the partitioning of ES-IS routing (ISO 9542 and ISO DP 10030) into supporting connection-less and connection-oriented communications is not discussed in ISO TR 9575.)

This article describes the OSI routing framework as stated in TR 9575 and gives a context for understanding the following articles on ISO routing protocols.

OSI Routing Framework

The OSI Routing Framework has four main sections: 1) Routing Concepts, 2) Environment for OSI Routing, 3) Goals for OSI Routing, and 4) the Structure of Global OSI Routing. The fourth section is our main interest. However, lets briefly mention the other three.

The Routing Concepts section discusses the main functions in routing. They include the Routing Information Bases, information collection and distribution, route calculation, and packet forwarding.

The Environment for OSI Routing discusses various interconnection possibilities. These include LAN interconnection, interconnection of private and public networks, campus and factory networks, and multi-vendor networks.

The Goals for OSI Routing section spell out general principles and requirements. In particular, they 1) allow for multiple subnetwork types (LANs, X.25 nets, etc.); 2) accommodate very large numbers of ESs and ISs; 3) argue for simple ESs; 4) allow for multiple organizations therefore requiring distribution of control, trust, firewalls, and security; and 5) allow use of existing network layer protocols.

Four functional routing tiers

The fourth main section describes an architecture for OSI Routing. In particular, it divides routing into four functional tiers: 1) routing between ESs and ISs, 2) routing between ISs within a Routing Domain, 3) routing between ISs in different Routing Domains but within an Administrative Domain, and 4) routing between ISs in different Administrative Domains.

We partition routing between ESs and ISs (the first functional tier) and between ISs (tiers 2 through 4) because there are fundamental differences between what an ES must know to route a packet, and what and IS must know. ESs do not forward packets on behalf of other systems, while ISs do. Therefore, ESs can be viewed as stubs hanging off of one or more ISs—essentially a simple pairwise binding between ESs and ISs. ISs, on the other hand, must be viewed of as part of a larger topology. More (and more complex) information about ISs must be distributed and collected. It is important that ESs be kept simple (this is one of the goals). Since there are many more ESs than ISs, it makes sense to put the greater complexity in fewer systems if possible. Further, if routing protocols must be changed (for instance, as they evolve or as networks grow), it is better if only ISs need be changed.

Before we can discuss the remaining three tiers, we must define Routing Domain and Administrative Domain. A *Routing Domain* (RD) is a set of ISs (and probably ESs as well, although this is not a critical distinction) that run a single instance of an Intra-domain IS-IS routing protocol. An *Administrative Domain* (AD) is a set of ISs that belong to and are administered by a single administration (in particular, the administration determines what routing protocols will be run in its ISs, and what its relationship will be with other Administrative Domains).

Administrative Domains

The purpose of Administrative Domains should be clear. The owners and administrators of a network want to determine autonomously how their network(s) will operate. They want to be protected from failures in other administrations' networks. Still, they may want to communicate with systems in other ADs, and may even be willing to forward traffic between two other ADs.

Routing Domains

The purpose of Routing Domains is to allow for the operation of different IS-IS protocols (or instances thereof) within a single AD. For example, a corporate internetwork may consist of some factory networks and some campus-style networks. For performance reasons, the factory networks may require different routing protocols than the campus-style networks. Probably more often than not, Routing Domain boundaries will co-incide with Administrative Domains—each AD will consist of one RD.

Routing at functional tier 2 (routing between ISs within a Routing Domain) is more concerned with performance (quick convergence) than with firewalls or policy issues. Routing at tier 4 (routing between ISs in different Administrative Domains) is more concerned with firewalls, security, and policy considerations than with performance. It seems likely that many administrations will forego the dynamic exchange of routing information altogether in the interest of security and firewalls.

Requirements for routing at tier 3 fall somewhere between those for tier 2 and those for tier 4. At this time, it appears unlikely that a separate protocol will exist for tier 3 routing. In fact, the inter-domain routing protocols introduced so far don't even distinguish between Administrative Domains and Routing Domains in their titles. The Intra-Domain IS-IS routing protocol currently moving through ISO is for routing within an RD (and therefore is also for routing within an AD). The only Inter-Domain IS-IS routing protocol so far introduced operates at both tiers 3 and 4.

Components of OSI: IS-IS Intra-Domain Routing

by Paul Tsuchiya, MITRE Corporation

Introduction

Currently, ANSI (the American National Standards Institute) X3S3.3 (Network and Transport layers) is pushing an intra-domain IS-IS routing protocol through ISO. The title of the protocol is "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol" (IS-IS for short). IS-IS is based on Digital Equipment Corporation's (DEC) Phase V routing.

At this time, IS-IS is not even at Draft Proposal (DP) status in ISO, and therefore doesn't have an ISO standard number yet. It is very likely that IS-IS will become a DP in January of 1990.

This article gives a broad overview of IS-IS, giving the major aspects of IS-IS that distinguish it from other routing protocols without getting into engineering details. Those are available in the IS-IS protocol specification.

Goals

IS-IS is meant to run within a *Routing Domain* (see the article on OSI Routing Architecture, page 38). This means that small convergence time and simplicity of operation are greater concerns than trust and firewalls. The main design goals of IS-IS are:

- *Large Size*: IS-IS is designed to handle 100,000 NSAPs (Network Service Access Points, which can be thought as ES addresses), and 10,000 ISs.

- *Convergence*: It converges to correct routes quickly without oscillation.

- *Simplicity*: In particular, ESs are simple, but the configuration and parameter tuning of ISs is also relatively simple. It is also relatively simple to maintain.

- *Multiple Subnetwork Types*: IS-IS is designed to operate over multiple subnetwork types, such as LANs, point-to-point links, and X.25 subnetworks.

To accommodate the large size, IS-IS is organized into a two-level hierarchy. To converge quickly, IS-IS uses a link-state routing algorithm (the same type that the Arpanet SPF routing algorithm uses). Using link-state prevents the count-to-infinity type of oscillations seen in poorly designed distance-vector algorithms such as RIP. IS-IS uses non-traffic-based metrics, thereby avoiding traffic oscillations. Once addresses and metrics are configured, IS-IS runs automatically: routes are automatically calculated and recalculated upon topology change without manual intervention.

Link State

As already stated, IS-IS uses a link-state routing algorithm. The idea behind link-state is that each IS obtains a full topology map of the network (which ISs and ESs are connected to which other ISs and ESs). Using this map, each IS can generate a next-hop forwarding table by calculating the shortest path to all ESs. As long as all ISs have identical topology maps, and use the same algorithm to generate the next-hop forwarding table, correct routing will result.

The topology maps (one in each IS) are distributed and collected using flooding. Each IS generates a link-state update consisting of the list of ESs and ISs it is connected to, along with the metric associated with the link. The link-state update is flooded to all neighbor ISs, who flood it to their neighbor ISs, and so on. Sequence numbers are used to terminate the flood, and to distinguish old link state updates from new ones. Since every IS will receive link-state updates from every other IS, every ISs can build a full topology database. When the connectivity of an IS changes, it floods another link-state update.

Two-level Hierarchy

IS-IS is a two-level hierarchy. It classifies its ISs as level 1 and level 2 routers (the term router here is synonymous with IS). Level 1 routers are lower in the hierarchy, and form level 1 areas. Level 2 routers are higher, and provide routing between level 1 areas (see Figure 1).

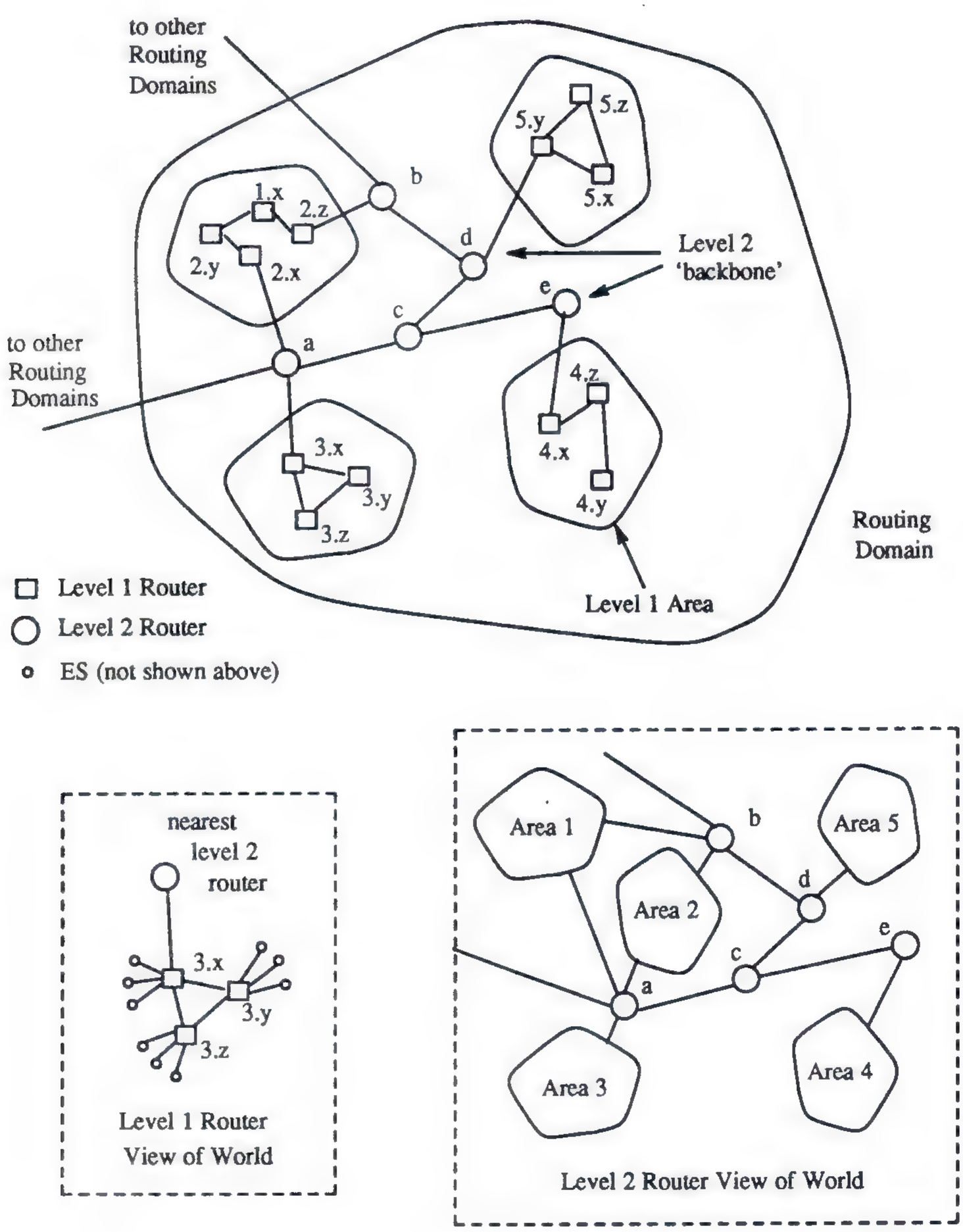


Figure 1: Routing Hierarchy

continued on next page

OSI IS-IS Intra-Domain Routing *(continued)*

Notice that all the level 2 routers are connected, that is, any level 2 router can reach any other level 2 router by going through only level 2 routers. Level 2 routers form a backbone that connects different level 1 areas. While hierarchies can be built without this sort of backbone configuration, doing so simplifies the operation of both level 2 and level 1 routers.

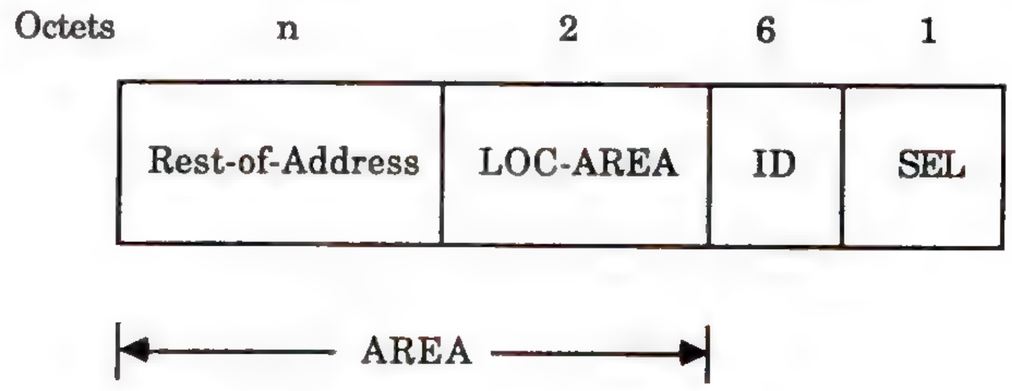
The simplification is mainly because the connection of level 2 routers frees level 1 routers from knowing anything more than how to route to the nearest level 2 router (see the left-hand box, Figure 1). Once a packet reaches a level 2 router, it can reach its destination level 1 area (or other Routing Domain) via level 2 routers exclusively. If this were not true, then level 1 routers would need to know the location of all other level 1 areas to prevent loops as the packet was sent from level 2 routers to level 1 routers and back to level 2 routers.

Another advantage of the connected level 2 "backbone" is that it simplifies routing protocol evolution. Since level 1 routers need not know the level 2 routing protocol or topology, changes can occur at level 2, or in other level 1 areas, without affecting the operation of other level 1 routers.

Two penalties for this simplification are 1) a restriction in topology, and 2) increased path lengths. The topology restriction is slight, however, considering that 1) many Routing Domains have a backbone topology anyway, and 2) any level 1 router can be administratively upgraded to a level 2 router if necessary to connect the backbone. The increase in path length is best described by example. Assume a message from an ES attached to level 1 router 1.x is destined to an ES attached to 3.x. Since 1.x will forward the packet to its nearest level 2 router, the packet will go to router b via 2.z (assuming that all links in Figure 1 have the same routing metric value). Router b will forward it over the level 2 backbone to router a, which will deliver it to 3.x. The path chosen is six hops, two hops more that the shortest path via 2.y.

Addressing

The addresses in IS-IS reflect the two-level hierarchy. The suggested address format in IS-IS is:



The SEL field selects the transport protocol machine (for instance, OSI Transport vs. TCP). The ID field identifies an ES. The ID must be unique among all NSAPs in the same level 1 area. The ID need not be globally unique, and in particular need not be a MAC address. However, it is convenient for the network administrator to assign MAC addresses into the ID field to ensure uniqueness. The LOC-AREA identifies the area that the ES (or more accurately, NSAP) is in, and must be unique among NSAPs with the same Rest-of-Address. The Rest-of-Address and LOC-AREA together make up the AREA. It is AREA that level 2 routers look at to make the level 2 routing decision.

This address structure reflects the two-level hierarchy. Level 1 routers only need know of the ESs and other level 1 routers in their level 1 area, and of the nearest level 2 router (lower left-hand box in Figure 1). Level 2 routers need only know of other level 2 routers in their Routing Domain, the location of level 1 areas, and the best exit level 2 router for other Routing Domains (lower right hand box in Figure 1). The existence of and distance to other Routing Domains is administratively configured into level 2 routers that share links with routers in other Routing Domains. The other Routing Domains are included in level 2 routing updates and spread to all other level 2 routers. When an inter-domain routing protocol exists, it may be possible for border level 2 routers to learn about other Routing Domains dynamically.

In Figure 1, the numbered part of the address to the left of the dot represents the AREA, while the lettered part of the address to the right of the dot represents the ID. I have not bothered to label each ES's address. Notice that one of the areas has level 1 routers with different area addresses (1 and 2). This is seen by level 2 routers as two separate areas (but reachable via the same routers). IS-IS allows multiple AREA address fields in the same area. This eases the assignment of addresses because it allows two areas to be merged, or the simple addition of ESs to an area, particularly those with a different Rest-of-Address. However, it is more efficient if all ESs in an area have the same AREA address.

Area Partitions

One of the problems associated with an area hierarchy is the possibility of an area partition. For instance, if the link between 2.y and 1.x in Figure 1 broke, then level 1 router 2.y could not deliver packets to 2.x even though a physical path exists (via level 2 routers). This is because, on the one hand, it would think that 2.x was in its area because of the AREA part of the address and therefore not route it to a level 2 router, but on the other hand would not be able to route the packet within the area because of the partition. Further, some packets coming from the backbone would be misdelivered because they would enter the wrong partition segment. We call this a *level 1 partition*. Level 2 partitions are also possible. For instance, if the link between level 2 routers c and d broke, then packets coming from area 3 would not get to area 5 even though a physical path through area 2 exists.

IS-IS dynamically repairs level 1 partitions, but not level 2 partitions (I am not sure why both types are not handled). The partition repair mechanism works entirely in level 2 routers. Level 2 routers discover the partition because they get inconsistent information from 1) level 1 routers telling which level 2 routers are attached to an area, and 2) from level 2 routers telling which level 1 areas can be reached. (Note that all level 2 routers that are attached to an area are also level 1 routers. This gives them access to the level 1 router information required to know which ESs are in their partition segment.) When a partition is discovered, the lowest numbered level 2 router in each partition segment executes the repair. They do this by establishing a virtual level 1 link between them via level 2 routers, and passing level 1 routing updates over the level 1 virtual link. The mechanism for establishing the virtual level 1 link is encapsulation. The level 1 routers don't know that the virtual level 1 link passes through level 2 routers—they see it as no different from real level 1 links.

continued on next page

OSI IS-IS Intra-Domain Routing *(continued)*

For example, consider again the case where the level 1 link between 2.y and 1.x goes down. Level 2 routers a and b will discover the partition, establish a virtual link between them, and pass level 1 router updates between them. Now assume that an ES attached to level 1 router 4.x has a packet for an ES attached to level 1 router 2.z. The packet will initially be routed to level 2 router a since it is the shortest path to that area (again assuming that all links shown have identical metric values). Level 2 router a will recognize that it cannot deliver the packet via level 1 links, and so will encapsulate the packet in a CLNP (ISO 8473) header addressed to level 2 router b. The packet will backtrack to level 2 router b, who will decapsulate the packet and deliver it to level 1 router 2.z.

Notice that no routers other than level 2 routers a and b were even aware that a partition existed and was repaired. The dual penalties for the advantage of localizing the repair effort are sub-optimal paths and the extra burden of encapsulation and decapsulation. Clearly the idea is here is that the link that caused the partition be fixed relatively soon.

Metrics

In the examples, I have assumed for illustrative simplicity that all metric values are the same. Of course, metric values should be assigned to control the flow of traffic over links. Currently IS-IS has one metric. It is a one-octet parameter. The length of a path is the sum of the metric values for the links on that path. One octet was chosen because it gives adequate granularity on one hand, but allows for an efficient implementation of the shortest path algorithm on the other. (The shortest path algorithm is the one that generates the forwarding table from the topology map. In the IS-IS standard, it is called simply the *routing algorithm* or *route calculation*. The algorithm used is attributed to Dijkstra, and is also called SPF for *Shortest Path First*.) The implementation of the shortest path algorithm described in IS-IS assumes that an intermediate table of entries ordered by distance is kept. By limiting distance values to one octet, this intermediate table is limited to a reasonable size.

Recently there has been contributions from the international community suggesting that multiple metric types be allowed—at least those encoded in the ISO 8473 CLNP header (residual error, cost, and transit delay). These suggestions are being seriously considered, but as of yet no decision has been made.

Partial routing updates

Another improvement that has recently been added to IS-IS (but has not yet been published in the standard) is that of *partial updates*. The problem here is of keeping routing updates small while still keeping track of which neighbors are reachable and which are not. The routing update not only conveys which neighbors are reachable (by listing them), it also conveys which neighbors are not reachable (by not listing them). Therefore, if a neighbor was listed in a previous routing update but is not in the latest routing update, that neighbor is assumed unreachable. As a result, all existing neighbors are listed in all routing updates.

If a router has a large number of neighbors, which can happen for instance when many routers are attached to a broadcast LAN, then the routing updates can be large. This requires fragmentation, and consequently slows down the entire update process because the fragments must be assembled at each hop.

One possible solution, called *incremental updates*, is to no longer assume that the absence of a neighbor in a routing update indicates that it is unreachable. Instead, routing updates explicitly list the reachability status of neighbors, and anything not listed is assumed to have not changed status. The problem with this is that if a routing update is not received correctly, then the database is permanently out of sync. For instance, if a router does not receive a transmitted routing update stating that a neighbor is unreachable, then it will continue to assume that the neighbor is reachable even though other routers know otherwise. No event (except the neighbor becoming reachable again) will fix the discrepancy because the routers on either end of the (now unreachable) neighbor will have no reason to send routing updates about that neighbor. In particular, periodic routing updates to keep the data bases in sync are not practical because the routing update would have to contain the status of all current and previous neighbors—just what we were trying to avoid.

With partial updates now used in IS-IS, the otherwise full update is split into multiple updates, and each update indicates the numerical range of neighbors that are considered by the update. This way, the semantics of explicitly listing reachable neighbors and implicitly listing unreachable neighbors is preserved. If a partial routing update does not list a neighbor that is in the range indicated by the update, then it is assumed that the neighbor is unreachable. If a partial update is received incorrectly, then the next periodically sent partial update will re-synchronize the databases. The various partial updates each maintain their own sequence number. If the status for a neighbor changes, only the partial update containing that neighbor need be resent.

Pseudo-Nodes

Another method of limiting the size of routing updates is that of *pseudo-nodes*. The problem happens if there are many routers on a broadcast LAN. In the general sense, each is a neighbor of the others. If they all sent full routing updates, then N routers would each send a routing update with N-1 neighbors, thus burdening the network with order N(N-1) information. To prevent this, one router on the LAN is elected as the LAN pseudo-node. It advertises all other routers (including itself) as its neighbors, but all other routers only advertise the pseudo-node as their neighbor. This way, a star configuration with N links is formed, rather than the fully-connected configuration with N(N-1) links. To maintain the correct metric values, the metric value from the pseudo-node to the (real) routers is zero.

Operation over 8208 subnetworks

When operating over an 8208 (X.25) subnetwork, IS-IS distinguishes between circuits set-up by administrative procedure (static) and circuits set-up because of the receipt of a PDU (dynamic). Of the dynamic circuits, IS-IS also distinguishes between those for which the NSAPs reachable via the neighbor are learned from routing updates, and those for which the NSAPs reachable via the neighbor are locally administered. This latter case allows the use of 8208 subnetworks without requiring circuits (which are expensive) be used for exchanging routing updates.

PAUL TSUCHIYA is a Member Technical Staff at the MITRE Corporation. He is currently researching algorithms for dynamic routing in very large networks called Landmark Routing. Paul is a member of the IETF Open Routing Working Group, the Autonets Task Force, and is Vice Chairman of ANSI X3S3.3 ISO Network and Transport Layer Group.

Components of OSI: ES–IS Routing

by Rob Hagens, University of Wisconsin

Introduction

On a typical OSI subnetwork, computers (systems) generally fall into two classes: *End Systems* (ES) and *Intermediate Systems* (IS). This classification is made from a layer point of view: both systems contain the OSI layers 1 (physical) through 3 (network). However, an ES also contains OSI layers 4–7. An IS does not. (Note: The distinction between ESs and ISs is gray when real systems are considered. For example, although an IS may primarily route packets, it may also have layers 4–7 present in order to operate network management protocols.)

The basic idea is that ESs are systems that run applications that send and receive packets of information and ISs are systems that route the packets from source to destination. This general concept can be readily subdivided into two cases. The first case occurs when the source and destination ES are connected to the same subnetwork (see Figure 1, host A communicating with host B). Since A and B are on the same subnetwork, packets may flow directly between them without the intervention of an IS. The second case occurs when the source and destination ES are not connected to the same subnetwork (Figure 1, host A communicating with host C). It is important to note that in the second case, packets flowing from A to C must be forwarded from one subnetwork to the other by IS D.

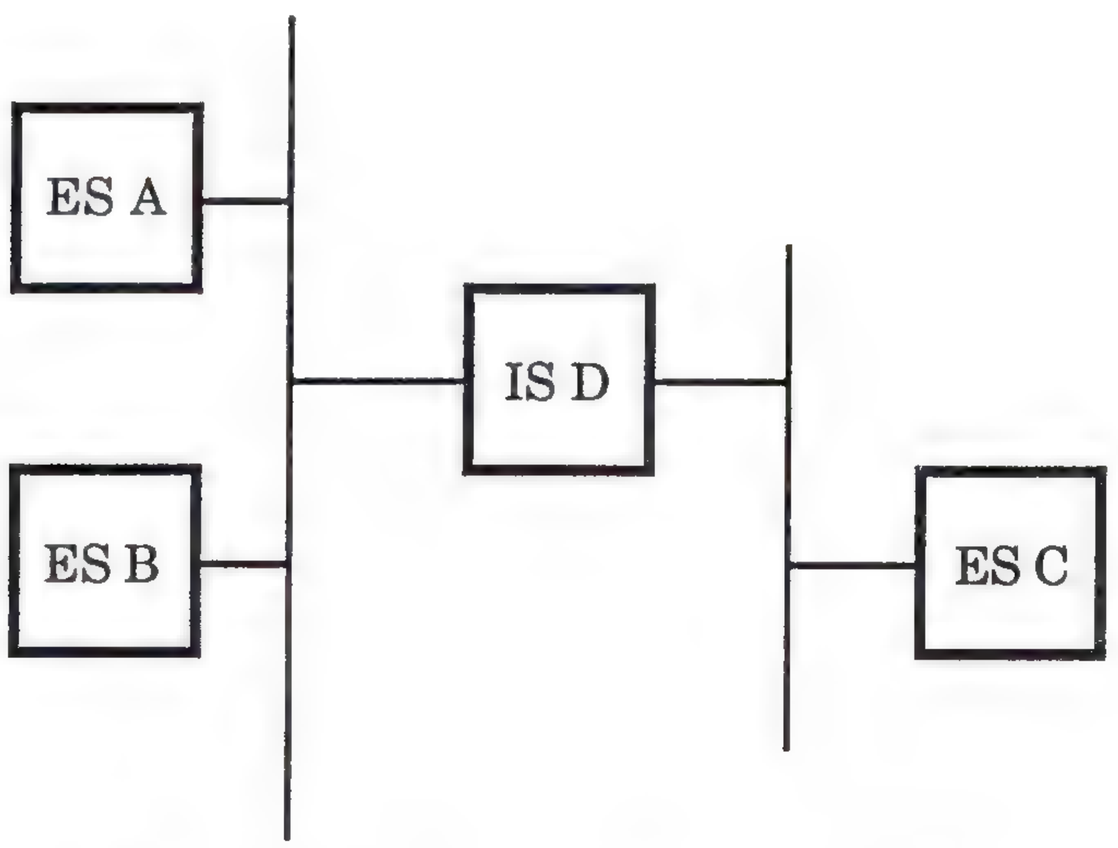


Figure 1: Example network configuration

The *End System to Intermediate System Routing Exchange Protocol* (ISO 9542) is designed to aid the two types of communication between ESs and ISs described above. The protocol provides a means for ESs and ISs on a subnetwork to learn of each other's existence. This process, known as *configuration*, allows ESs and ISs to dynamically determine the existence of each other. The dynamic quality of the protocol eliminates the need for manual configuration of the different types of systems on a subnetwork. In addition to promoting the exchange of configuration information, the protocol provides a means for route *redirection* information to be sent from an IS to an ES. This allows an IS to inform an ES of a potentially better route towards a destination.

As explained below, the ES-IS protocol distinguishes different types of subnetworks. The operation of the protocol is quite different on each type of subnetwork. The difference between the broadcast subnetwork (i.e., 802.3) version and the general topology subnetwork (i.e., X.25) version is so large that there are two distinct versions of the protocol defined. This article will not describe the operation of ES-IS over X.25. Rather, it will concern itself with the operation of the ES-IS protocol when it is used in conjunction with the OSI connectionless-mode network protocol (ISO 8473).

Configuration information

Configuration information is transmitted by exchanging two types of packets: the *End System Hello* (ESH) packet is generated by ESs and transmitted to every IS on the subnetwork. The *Intermediate System Hello* (ISH) packet is generated by ISs and transmitted to every ES on the subnetwork. The primary purpose of the Hello packets is to convey the subnetwork and network layer address of the system that transmitted the packet. The ESH and the ISH packets are generated by a system when the system's configuration timer expires. When the timer expires, a Hello packet is transmitted and the timer reset to its initial value. The initial value of the timer is maintained locally by the system. The interval between transmission of Hello packets may be modified by adjusting this timer.

The configuration interval must be coordinated with the value of the holding timer that is transmitted with the Hello packet. The holding timer indicates to the receiving system the number of seconds that the configuration information should be kept. When the holding timer expires, the associated configuration information must be deleted. If the holding timer is smaller than the configuration timer, systems may "disappear" from the subnetwork for periods of time.

Redirection information

Redirection information is transmitted via an RD PDU. These PDUs are only generated by ISs. The purpose of an RD PDU is to inform an ES of a potentially better route towards its destination.

During the forwarding process, an IS will determine the next system to which the packet should be sent. If an IS can determine that this "next hop" system can be reached directly, without the IS's intervention, then a RD PDU will be sent from the IS to the originator of the packet.

The RD PDU may redirect an ES to either the destination ES or to a different IS. The lifetime of the information conveyed by the RD PDU is limited by a holding timer, just like the Hello PDUs.

Normally, the redirection information delivered in an RD PDU is specific to a single destination address. However, as an option, it is possible for an RD PDU to indicate that the redirection information should be applied to a larger number of destination addresses. This is accomplished by including an address mask which indicates the larger population of destination addresses.

Subnetwork topology considerations

The ES-IS protocol identifies 3 common subnetwork topologies: a point-to-point subnetwork, a broadcast subnetwork, and a general topology subnetwork. The point-to-point subnetwork supports exactly two systems. The broadcast and general topology subnetworks support an arbitrary number of systems. The crucial difference between the latter two types is the cost of sending a packet to a large subset of the subnetwork population.

continued on next page

OSI ES–IS Routing *(continued)*

If the cost of such an *n-way* transmission scales directly with the subset size, then the subnetwork topology is considered general (a common example is an X.25 subnetwork). If the cost of an *n-way* transmission is close to the cost of a transmission to a single system, then the topology is broadcast (a common example is 802.3). This difference affects the transmission of configuration information.

Ideally, configuration information is sent to many systems on the subnetwork simultaneously. When operating on a broadcast subnetwork, ESHs are sent to a special multicast address "all intermediate systems." ISHs are sent to a special multicast address "all end systems." This multicast address may be realized physically as a true multicast or as a broadcast to every system. When operating on a general topology subnetwork, configuration information is generally not transmitted due to the high cost of a multicast transmission.

Network and Subnetwork addresses

The addresses described so far have been network layer addresses. There are actually two different kinds of addresses that the ES-IS protocol conveys: *network layer addresses* and *subnet addresses*.

Network layer addresses identify either the interface between layer 3 and layer 4 in an OSI ES (called a Network Service Access Point, NSAP), or the network layer entity itself in an OSI IS (called a *Network Entity Title*, NET). NSAP addresses and NETs are part of a name space which spans all OSI networks.

These network layer addresses can be contrasted with a subnetwork point of attachment address (SNPA address). The SNPA is the point where an ES or IS is physically attached to a subnetwork. The SNPA address is used by the subnetwork to uniquely identify each system attached to the subnetwork. For example, in an 802.3 subnetwork, the SNPA address corresponds to the 48 bit MAC address.

When an ES or IS transmits a packet, it is necessary to determine the destination SNPA address. This is generally accomplished by associating a NSAP address or NET with a specific SNPA address. Part of the configuration information transmitted by the ES-IS protocol is this NSAP address (or NET) to SNPA address mapping.

Operational scenarios

It is often useful to consider real-life scenarios in order to get an intuitive idea of how a protocol operates. In the following examples, please refer to Figure 1. Furthermore, assume that the subnetworks in Figure 1 are broadcast subnetworks and that capital letters are NSAP & NET addresses, whereas lower letters are corresponding SNPA addresses.

Example 1: ES A wishes to send a packet to ES C. ES A knows about IS D because it receives periodic ISH packets from IS D. Since D is an IS, and ISs should know how to route packets toward any destination, ES A will send its packet directly to IS D. In addition to knowing that IS D is an intermediate system, ES A also knows the SNPA for IS D. This information was taken from the ISH received from IS D. Result: ES A sends the packet to IS D at SNPA d and IS D forwards the packet to ES C.

Example 2a: ES A wishes to send a packet to ES B. As in example 1, ES A knows about IS D due to receipt of ISH packets from D. Therefore, ES A will send its packet to IS D at SNPA d. IS D has knowledge about both ES A and ES B because it receives ESH packets from both. When IS D receives the data packet from ES A, it computes that ES B is reachable via the same subnetwork as ES A, and forwards the data packet to ES B. In addition, IS D computes that ES A could have sent the packet directly to ES B. IS D notifies ES A of this fact by sending a RD PDU to ES A. This RD PDU indicates that ES B may be reached by sending packets directly to SNPA b. Further packets sent by ES A to ES B are directed toward ES B.

Example 2b: This is the same scenario as 2a, but with a twist. Assume that ES A's routing information states that ES B is on the *same subnetwork* as ES A. The only problem is that ES A does not know the SNPA for ES B. ES A has two choices. As in 2a, it could send the packet to IS D and receive a redirect. Or, as an optimization, ES A could listen to ESH PDUs sent on the subnetwork. If ES A eavesdropped, it would know the SNPA of B. ES A could then transmit the packet directly to ES B at SNPA b.

Example 3: Similar to scenario 2b, assume that ES A wishes to send to ES B. In addition, ES A knows that ES B is on the same subnetwork. Furthermore, assume that ES A does not eavesdrop and IS D does not exist. How can ES A determine the SNPA of ES B? In this situation, ES A will invoke a function of the ES-IS protocol called query configuration. To do this, ES A will *broadcast* its *data* packet to all ESs on the subnetwork. When ES B receives this broadcasted packet, it will note: 1) that the packet is a data packet sent to a broadcast SNPA address; 2) that the destination NSAP address of the broadcasted packet is ES B. In this case, ES B will generate an ESH packet and send it directly to ES A. This packet will inform ES A of the SNPA of ES B.

A commonly asked question, when considering example 3, is "If ES A does not eavesdrop ESH packets, how can it receive the ESH sent directly by ES B ?" The key to the answer is SNPA addresses. When an ES eavesdrops, it *enables* the reception of packets addressed to the multicast address "all intermediate systems" (because end systems send their ESH packets to the multicast address "all intermediate systems"). When ES B sends an ESH to ES A, it sends it directly to SNPA a. ES A will receive this packet even if the "all intermediate systems" multicast address is not enabled.

PDU formats

Figure 2 and 3 show the format of the ESH and ISH packets. The format of these packets is very similar: they both have a fixed header which is almost identical to the fixed part of an ISO 8473 (CLNP) packet. Following the fixed part, each Hello packet has space to store a network address. Note that an ESH has space for more than 1 NSAP address, (since an ES may support several NSAPs) whereas an ISH has space for only 1 NET.

The discerning reader will note that there is no place in either the ESH or ISH packet to store an SNPA address. How can this information be transmitted from system to system? The answer lies in the subnetwork service definition. By fiat, when a packet is delivered by a subnetwork service, the delivery will include not only the data packet but also the subnetwork source and destination address.

continued on next page

OSI ES–IS Routing *(continued)*

Thus, when processing an ESH, the SNPA associated with the ES is taken from the *source SNPA address* which accompanies the delivery of the ESH.

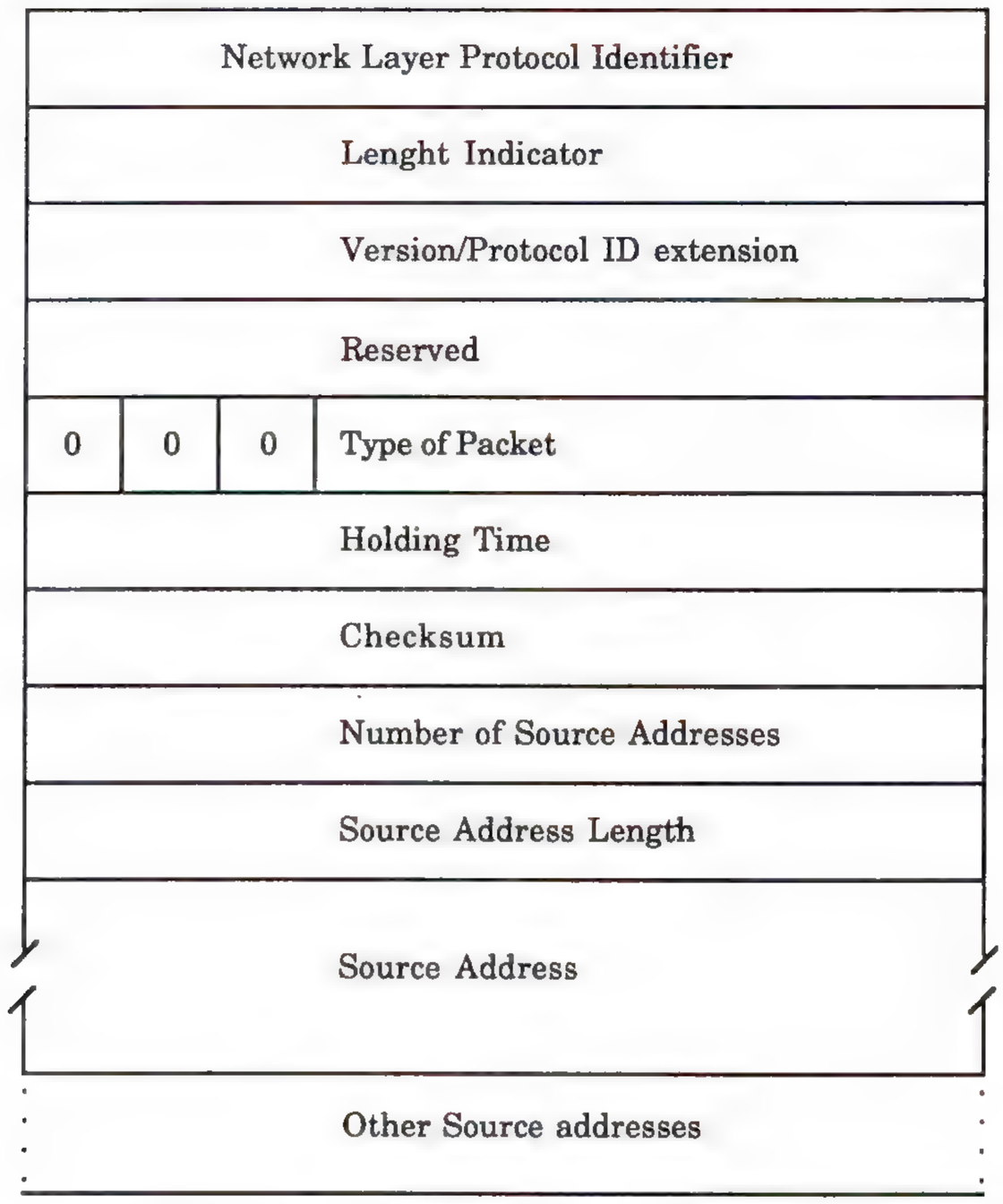


Figure 2: The ESH Packet Format

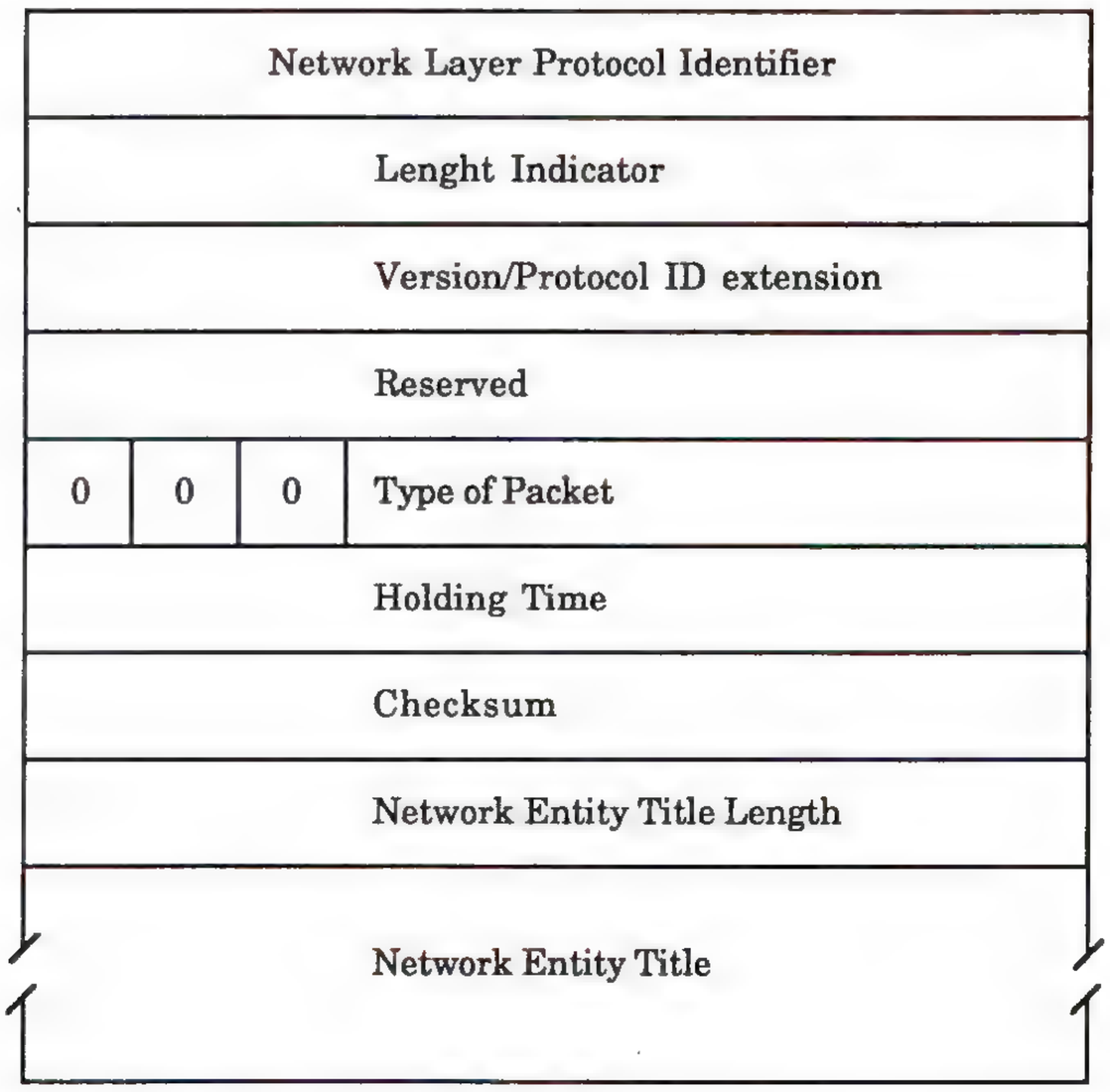


Figure 3: The ISH Packet Format

Comparison to DoD protocols

The ES-IS routing exchange protocol does not have an exact counterpart in the DoD protocol suite. The configuration information exchange function is similar to the function of the Address Resolution Protocol (ARP, RFC 826). However, ARP was designed around an "on demand" approach, that is, query when the information is needed. This is in contrast to ES-IS where the design is based around continual update messages (Hello packets). It should be noted that the query configuration function (shown in example 3) is somewhat analogous to the ARP "on demand" approach. However, unlike ARP, query configuration requires that the entire data packet, rather than an ARP request, be broadcast on the subnetwork.

The counterpart to the ES-IS redirection information function in the DoD protocol suite is the redirect message defined by the ICMP protocol (RFC 792). These two approaches to redirect differ in that ES-IS allows the target of a redirect to be either an ES or IS whereas an ICMP redirect can only refer to a gateway.

ROBERT A. HAGENS received an M.S. in Computer Science from the University of Wisconsin-Madison in 1984. Hagens is currently an Associate Researcher in Network Communications at the University of Wisconsin-Madison. His current work interests are in user interface design, especially when related to electronic mail. He is also the co-chair of the IETF OSI working group. From 1985 to 1988 he implemented the ISO Session and Network layer protocols and provided UNIX kernel systems support for the Wisconsin OSI project. In 1984, Hagens worked on the WISCNET project (DoD protocol suite for IBM VM/SP systems) including implementation and performance improvements to FTP, UDP, and TCP/IP.

Erratum

Even with the most sophisticated spell checkers, you sometimes end up with copy which doesn't quite read the way it was intended to read. In Last month's issue (Volume 3, No. 7, July 1989) the word *contract* was used instead of the word *contrast*. The last sentence of Nancy Hall's article on the OSI Transport Layer should read:

"In constrast, by assuming a worst-case network service, and using a common network protocol (IP), TCP avoids these problems while achieving equivalent functionality."

New book on Network Management

Expert Systems Applications in Integrated Network Management is a collection of papers edited by Eric Ericson, Lisa Traeger Ericson, and Daniel Minoli. The publisher is Artech House, and the book will be available in September. For more information, contact Artech House at 1-800-225-9977, extension 4002.

The IETF directories at SRI-NIC.ARPA

The Internet Engineering Task Force announces the availability of two newly revised directories, the "IETF:" directory and the "INTERNET-DRAFTS:" directory on the SRI-NIC.ARPA host.

Purpose

The "IETF:" directory has been established as an aid to both veteran IETF members and newcomers. It is comprised of files containing: a general description of the IETF (history, organization, goals); a summary of active Working Groups within the IETF; IETF meeting dates/locations; upcoming meeting information and an associated RSVP form; the upcoming meeting agenda; and a README file with an overview of directory contents. In addition, individual files have been dedicated to each Working Group and their particular activities. These files contain respective Charters, Status Updates and Current Meeting Reports. The WG files are named in the following fashion:

```
<WGNAME>.charter
<WGNAME>.status
<WGNAME>.report
```

Naming Scheme

The "INTERNET-DRAFTS:" directory presents drafts for review and comment. It contains documents that will be submitted ultimately to the RFC Editor to be considered for publishing as RFCs. Comments are welcomed and should be addressed to the responsible persons whose names and email addresses are listed on the first page of the respective draft. Each Internet-Draft is placed in a separate file; the following standard naming scheme is used:

File Format	Naming Scheme
ASCII text	DRAFT-<TFNAME>-<WGNAME>-<ABBREVTITLE>-<REVNO>.TXT
PostScript	DRAFT-<TFNAME>-<WGNAME>-<ABBREVTITLE>-<REVNO>.PS
UNIX compressed ASCII	DRAFT-<TFNAME>-<WGNAME>-<ABBREVTITLE>-<REVNO>.TXTZ
UNIX compressed PostScript	DRAFT-<TFNAME>-<WGNAME>-<ABBREVTITLE>-<REVNO>.PSZ

If the document is not being authored in a Task Force, then the author's name will be substituted for the Working Group name (WGNAME) and an organizational affiliation will be submitted for the Task Force name (TFNAME). Example:

```
DRAFT-<ORG>-<AUTHORNAME>-<ABBREVTITLE>-<REVNO>.TXT
```

Recent Drafts

Drafts recently installed include the following:

DRAFT-IETF-HOSTREQ-HRLL-00.TXT or TXTZ	Requirements for Internet Hosts — Communication Layers, edited by Robert Braden for the Host Requirements Working Group, June 16, 1989.

DRAFT-IETF-HOSTREQ-HRUL-00.TXT or TXTZ	Requirements for Internet Hosts — Application Layer, edited by Robert Braden for the Host Requirements Working Group, May 22, 1989.
DRAFT-IETF-PPP-REQ-00.TXT	Requirements for an Internet Standard Point-to-Point Protocol, edited by Drew Perkins for the PT-PT Protocol Working Group, June 1989.
DRAFT-IETF-PPP-IPDATAGRAMSTX-00.TXT	The Point-to-Point Protocol (PPP): A Proposed Standard for the Transmission of IP Datagrams over Point-to-Point Links, edited by Drew Perkins for the PT-PT Protocol Working Group, June 1989.
DRAFT-UCL-KILLE-X400RFC822-00.TXT	Mapping between X.400 (1988) and RFC 822, Steve Kille, University College London, June 5, 1989.
DRAFT-IETF-TELNET-LINEMODE-00.TXT	Telnet Linemode Option, edited by David Borman for the Telnet Working Group, July 1989.
DRAFT-IETF-HRWG-MULTIHOMING-00.TXT	Multi-homed Hosts in an IP Network, John Lekashman, April 26, 1988.

The "INTERNET-DRAFTS:" directory also contains a README file and an Index-Abstract file to aid the reader in locating drafts.

Getting IETF files

Both the "IETF:" and "INTERNET-DRAFTS:" directories are available on-line at SRI-NIC.ARPA and can be accessed by anonymous FTP. The "dir" or "ls" command will permit a review of what files are available and the specific naming scheme to use for a successful anonymous FTP action. For more information, please contact: ietf-request@venera.isi.edu.

—Karen Bowers

Reminder: INTEROP 89

INTEROP™ 89 will be held in San Jose, CA, October 2–6, 1989. Send in your registration before September 1 to receive the discount rate.

The conference offers 17 tutorials and 35 conference sessions, plus an exhibition with over 100 vendors showing true multi-vendor interoperability on the Show and Tel-Net. Call 415-941-3399 for information.

Internet Resource Guide

Introduction

The NSF Network Service Center (NNSC) is in the process of publishing a guide to resources on the Internet. These resources include facilities such as super-computers, databases, libraries, or specialized programs on the Internet which are available to large numbers of users. Each entry in the guide will contain a description of the resource, a description of who may use the resource, the type of access supported (email, FTP, Telnet), and contact points for further information such as a phone number, email address, and name of a contact person where appropriate. We expect each listing to be approximately one page of text.

Electronic distribution

Our initial concept was that the guide would be published in hardcopy form to provide readers with the ability to flip manually through the pages. However, we also realized that a major difficulty in printing hardcopy materials is keeping the information accurate and up-to-date. We've therefore decided to use an electronic submission and distribution system. We're creating a mailing list of people who wish to receive the guide and we will distribute pages of the guide in PostScript form. Recipients can print the pages locally and insert them into a looseleaf binder, which they can update when they receive new material. An initial set of entries was mailed to the list in June, and we will periodically send out new and updated entries. If you want to join the list to receive the guide, please send a message to resource-guide-request@nnsc.nsf.net. Hardcopy versions will be sent by U.S. mail to people who can't be reached by email.

Submitting entries

Submissions describing resources should be sent to the NNSC and will be edited here prior to distribution. If you're part of an organization that maintains a facility that should be included in the guide, please fill out the template at the end of this announcement and send it to: resource-guide@nnsc.nsf.net. A sample entry can be obtain via anonymous FTP from nnsc.nsf.net. In the directory resource-guide, get the file sample.ps. If you know anyone who might want to make a contribution, please pass on this message or send us a note giving us the name of a contact person.

We need your help to make the guide a valuable tool for Internet users—its effectiveness will depend on the active participation of the community of resource providers. Thanks. *—NNSC Staff*

Internet Resource Guide Template

(1) Name of Resource
(2) USPS Address
(3) Email Address (please use a general or central address)
(4) Phone Number (please use a central phone number)
(5) Description of Resource
(6) Network Access
(7) Who Can Use the Resource/Restrictions (list guidelines)
(8) Miscellaneous Information:
- additional phone numbers
- names of individuals if appropriate

Network Management Tool Descriptions Wanted

Catalog

The Internet Engineering Task Force (IETF) is now requesting entries for a catalog of tools for managing TCP/IP internets. The catalog will be a compendium of the tools available to assist network managers in debugging and maintaining their networks. Network management tools and LAN management tools (other than breakout boxes) will included. The IETF intends to publish the catalog as a Request For Comments (RFC).

Entries describing public domain, copyrighted, and commercial tools are solicited. Vendors are encouraged to submit descriptions of their products. Descriptions from knowledgeable users of commercial and noncommercial tools are also welcome.

Tool descriptions

Tool descriptions should be objective, no more than two or three pages long, and should contain the following sections (where applicable):

(1) Tool name.

(2) Key-word list (for now, a best guess. A uniform key-word list is under development).

(3) Abstract: a brief description of the tool's purpose and characteristics.

(4) Mechanism: how the tool works.

(5) Caveats: warnings about the tool's impact on system performance, etc.

(6) Hardware requirements.

(7) Software requirements.

(8) Related analysis tools: post processors, data reduction tools, graphics support, etc.

(9) Availability and support: how to obtain the tool, and whether it is in the public domain, copyrighted, or commercially available. This will not include pricing information.

(10) Bugs and limitations.

An IETF review panel will edit all catalog entries. Though contributors will be given final authority to approve or disapprove revisions to their submissions, the IETF reserves the right to exclude entries that do not meet its editorial criteria. Editing of the catalog will be completed this November.

Send tool descriptions or requests for further information to:

Robert Stine
SPARTA, Inc.
7926 Jones Branch Drive, Suite 1070
McLean, VA 22102
Phone: (703) 448-0210 FAX: (703) 734-3323
E-mail: stine@sparta.com

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040
415-941-3399
FAX: 415-949-1779

Bulk Rate
U.S. POSTAGE
P A I D
SAN JOSE, CA
PERMIT NO. 1